

The Science of Information From Language to Black Holes

Course Guidebook

Professor Benjamin Schumacher
Kenyon College



PUBLISHED BY:

THE GREAT COURSES

Corporate Headquarters

4840 Westfields Boulevard, Suite 500

Chantilly, Virginia 20151-2299

Phone: 1-800-832-2412

Fax: 703-378-3819

www.thegreatcourses.com

Copyright © The Teaching Company, 2015

Printed in the United States of America

This book is in copyright. All rights reserved.

Without limiting the rights under copyright reserved above,
no part of this publication may be reproduced, stored in
or introduced into a retrieval system, or transmitted,
in any form, or by any means
(electronic, mechanical, photocopying, recording, or otherwise),
without the prior written permission of
The Teaching Company.



Benjamin Schumacher, Ph.D.

Professor of Physics
Kenyon College

Dr. Benjamin Schumacher is Professor of Physics at Kenyon College, where he has taught for 25 years. He received his B.A. from Hendrix College and his Ph.D. in Theoretical Physics from The University of Texas at Austin, where he was the last doctoral student of John Archibald Wheeler.

Dr. Schumacher is the author of numerous scientific papers and books, including *Physics in Spacetime: An Introduction to Special Relativity*, and is the coauthor of *Quantum Processes, Systems, and Information*. As one of the founders of quantum information theory, Dr. Schumacher introduced the term “qubit,” invented quantum data compression (also known as Schumacher compression), and established several fundamental results about the information capacity of quantum systems. For his contributions, he won the 2002 International Quantum Communication Award, the premier international prize in the field, and was named a Fellow of the American Physical Society. Besides quantum information theory, he has done physics research on black holes, thermodynamics, and statistical mechanics.

Dr. Schumacher has spent sabbaticals working at Los Alamos National Laboratory and the Institute for Quantum Information at the California Institute of Technology, where he was a Moore Distinguished Scholar. He also has conducted research at the Isaac Newton Institute for Mathematical Sciences of the University of Cambridge, the Santa Fe Institute, Perimeter Institute, The University of New Mexico, the University of Montreal, the University of Innsbruck, and The University of Queensland.

At Kenyon College, Dr. Schumacher teaches physics primarily, but he also regularly ventures into astronomy, mathematics, scientific computing, and the humanities.

For The Great Courses, Dr. Schumacher also has taught *Black Holes, Tides, and Curved Spacetime: Understanding Gravity*; *Quantum Mechanics: The Physics of the Microscopic World*; and *Impossible: Physics beyond the Edge*. ■

Table of Contents

INTRODUCTION

Professor Biography	i
Course Scope	1

LECTURE GUIDES

LECTURE 1

The Transformability of Information	4
---	---

LECTURE 2

Computation and Logic Gates	17
-----------------------------------	----

LECTURE 3

Measuring Information	26
-----------------------------	----

LECTURE 4

Entropy and the Average Surprise	34
--	----

LECTURE 5

Data Compression and Prefix-Free Codes	44
--	----

LECTURE 6

Encoding Images and Sounds	57
----------------------------------	----

LECTURE 7

Noise and Channel Capacity	69
----------------------------------	----

LECTURE 8

Error-Correcting Codes	82
------------------------------	----

LECTURE 9

Signals and Bandwidth	94
-----------------------------	----

LECTURE 10

Cryptography and Key Entropy	110
------------------------------------	-----

LECTURE 11

Cryptanalysis and Unraveling the Enigma	119
---	-----

LECTURE 12

Unbreakable Codes and Public Keys	130
---	-----

LECTURE 13

What Genetic Information Can Do	140
---------------------------------------	-----

LECTURE 14

Life's Origins and DNA Computing	152
--	-----

LECTURE 15

Neural Codes in the Brain.....	169
--------------------------------	-----

LECTURE 16

Entropy and Microstate Information.....	185
---	-----

LECTURE 17

Erasure Cost and Reversible Computing	198
---	-----

LECTURE 18

Horse Races and Stock Markets.....	213
------------------------------------	-----

LECTURE 19

Turing Machines and Algorithmic Information.....	226
--	-----

LECTURE 20

Uncomputable Functions and Incompleteness.....	239
--	-----

LECTURE 21

Qubits and Quantum Information	253
--------------------------------------	-----

LECTURE 22

Quantum Cryptography via Entanglement	266
---	-----

LECTURE 23

It from Bit: Physics from Information	281
---	-----

LECTURE 24

The Meaning of Information	293
----------------------------------	-----

SUPPLEMENTAL MATERIAL

Answers to Questions	309
Key Equations in the Science of Information	323
Glossary	332
Bibliography	347
Image Credits	354

The Science of Information

From Language to Black Holes

We live in the age of information. But what is information, and what are its laws? Claude Shannon's 1948 creation of *information theory* launched a revolution in both science and technology. The key property of information is its amazing transformability. Any single message can take on dozens of different physical forms, from marks on paper to radio waves, and any type of information can be encoded as a series of binary digits, or *bits*. Combined with revolutionary advances in electronics, Shannon's ideas changed the world.

At the heart of information theory is the *entropy* of an information source, which in the simplest case is the logarithm of the number of possible messages. Shannon's first fundamental theorem shows that the entropy equals the minimum number of bits needed in a binary code for the messages. This holds true even when the messages occur with very different probabilities, as with the letters of English. Shorter codewords are used for more common letters (as in Morse Code) to compress the data closer to the entropy limit. Music, image, and video require even more powerful data-compression techniques, based on how we perceive such data. Perceptual coding faithfully represents the human subjective experience of a sound or sight, rather than the exact mathematical details.

Information can be lost to errors in a noisy communication channel, but Shannon's second fundamental theorem guarantees that noise can be overcome by error-correcting codes. Such codes protect information in situations from a scratched compact disc to faint transmissions from spacecraft billions of kilometers distant. As Shannon showed, the information capacity of a continuous signal, such as an electrical voltage or a radio wave, is limited both by bandwidth (the range of frequencies available) and the signal-to-noise power ratio.

Cryptography is the branch of information science that deals with keeping secrets. Just as the shape of a physical key opens a locked door, so the “key” information of a cipher allows encoded text to be read. The history of cryptography has been a competition between increasingly sophisticated ciphers and increasingly powerful techniques for cryptanalysis. Because unbreakable codes are possible but often impractical, the security of modern public-key cryptography is based on the immense difficulty of certain computational problems.

Biological systems are profoundly informational. Genetic information in DNA represents the structures of proteins used by an organism. Molecular biologists have encoded other kinds of data into DNA and modified the genetic code to create a wider range of proteins. The earliest living things may have been self-replicating RNA molecules in the waters of the early Earth. The survival of their genetic information was dictated by the competition between error processes and natural selection. The brain, too, is a biological information system, in which we are beginning to understand the codes for neural messages and memory formation.

Information theory has an even deeper connection to thermodynamics, the physics of energy transformations. Thermodynamic entropy is exactly the information we lack about the microscopic details of a macroscopic system. James Clerk Maxwell’s famous demon thought experiment and Rolf Landauer’s principle of information erasure set fundamental limits on the waste heat produced by any computing machine.

Shannon’s information also has a surprising connection to gambling and mathematical finance, through the controversial log-optimal betting strategy.

Modern information theory has gone beyond Shannon in two ways. The first, algorithmic information theory, is based on computation rather than communication. The algorithmic entropy of a binary sequence is the length of the shortest computer program producing that sequence. This concept of information has remarkable connections to the unsolvable mathematical problems of Kurt Gödel and the uncomputable functions of Alan Turing.

Quantum information, the information carried by microscopic particles, has concepts and rules quite different from Shannon's theory. It is measured in qubits. Such information cannot be "cloned," but it allows new and more powerful types of computing. Quantum entanglement, an exclusive information relationship between qubits, can be used as a basis for perfectly secure quantum cryptography. The information aspects of quantum physics, and other information-related insights from black hole physics and cosmology, have inspired such physicists as John Wheeler to speculate on an information basis for all of physical reality: "it from bit."

Communication in the absence of a pre-arranged code—the problem of anti-cryptography—arises in problems ranging from storing nuclear waste to signaling extraterrestrial intelligences. Though these are speculative questions, they illuminate profound issues about information, context, and meaning—both among human beings and in the wider universe.

The Transformability of Information

We live in a revolutionary age: the age of information. Never before in history have we been able to acquire, record, communicate, and use so many different forms of information. Never before have we had access to such vast quantities of data of every kind. Never before have we been able to translate information from one form to another so easily and quickly. And, of course, never before have we had such a need to comprehend the concepts and principles of information.

Defining *Information*

- ▶ Information is a paradox. On the one hand, it is physical. Whenever we communicate or record information, we make use of a physical medium, such as sound, light, radio waves, electrical signals, or magnetic patterns. On the other hand, information is abstract. The messages carried by our physical signals are not identical to the signals themselves.
- ▶ That dichotomy is the key to the amazing transformability of information: from letter to laser pulse, from radio signal to sound wave, yet somehow remaining the same.
- ▶ Perhaps the greatest example of information transformability in all of human history is the invention that gave us human history in the first place: writing.
 - Human speech is a pattern of sound. Spoken words travel a short distance, then fade away. But in writing, these spoken words are rendered as patterns inscribed on a surface. In this form, they become durable rather than ephemeral.

ROSETTA STONE

[illegible]

- Writing represents speech, and in alphabetic writing, each symbol represents a sound. But the rule of association between the symbol and the message—the code—is entirely arbitrary. There is no necessary connection between one set of scratches and the sound /p/. Writing works because both writer and reader know the code. Without the code, there is no way to understand what the symbols mean.
- ▶ The essential concepts of information—the concepts of message, symbol, and code—are not new in human experience. What is new is the jaw-dropping multiplicity of uses to which we have put those concepts in the age of information.

Claude Shannon and Information Theory

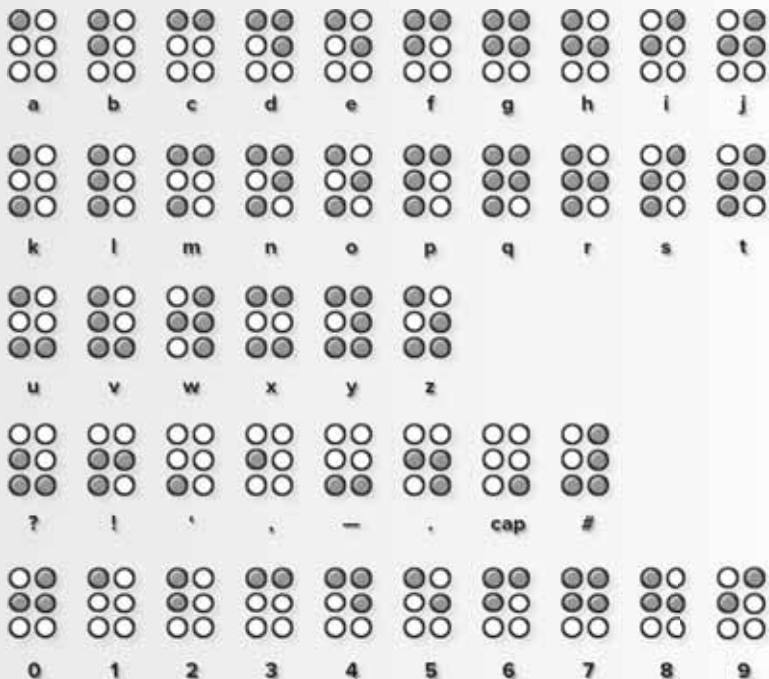
- ▶ In 1948, the American engineer and mathematician Claude Shannon published a paper that set forth the elements of a new science called *information theory*.
- ▶ Shannon first realized that the mathematical idea of information is not the same thing as meaning. The essential communication process is the same no matter what the message means.
- ▶ In the same way, the idea of information is not about the value or significance of a message. Meaning, value, and significance are obviously important qualities, but they are not the keys to understanding what information is.
- ▶ What's left, according to Shannon, is the distinction between different messages. Every message that is communicated is one particular message out of a set of many possible messages. Thus, Shannon's definition of information is as follows: "Information is the ability to distinguish reliably among possible alternatives."

Bits: Universal Information Currency

- ▶ If information is all about the distinction between possible messages, then there is a fundamental “atom” of information—the binary distinction between just two possible messages.
 - The messages might be yes and no for a simple question, on and off for an electrical signal, or the binary digits 1 and 0. It doesn’t matter what the messages mean; it only matters that there are two possible alternatives.
 - Such a simple two-value message is called a **bit**, meaning “binary digit.”
- ▶ Shannon pointed out that bits form a kind of universal currency for information. Our basic “alphabet” needs only two distinct symbols. And because we can transform information—freely switch from one code to another—any sort of information, such as numbers, words, or pictures, can be represented by bits: arrangements of binary digits, 0s and 1s.
- ▶ A concrete example is the code that was established for teletype machines. In this code, each letter of the alphabet is represented by 5 bits. The letter A is 11000, the letter B is 10011, a space between letters is 00100, and so on.
- ▶ Why does the teletype code use 5 bits? The answer involves a fundamental fact of information theory.
 - A code represents different messages by a series or string of symbols—in this case, the five binary digits. This is the *codeword* that represents the message. But the code must preserve the information, that is, the distinction between messages, so that no two different messages can be represented by the same codeword.
 - That means that the number of possible codewords in the code can be no smaller than the number of possible messages (designated M). In other words: (# of possible codewords) $\geq M$. That’s our fundamental fact about codes.

- In the teletype, we're coding letters of the alphabet, of which there are 26. Because we also need a character for the space between words, we can suppose that $M = 27$. How many 5-bit codewords do we have?
 - There are two possible values for the first bit, and for each of those, two possible values for the second bit, and so on; thus, the total number of possible 5-bit combinations is $2 \times 2 \times 2 \times 2 \times 2 = 2^5 = 32$.
 - The fact that the number of codewords (32) is greater than the number of possible messages (27) means that the 5-bit system works. If we tried to get by with only 4 bits, we would have only $2^4 = 16$ available codewords, which wouldn't be enough.

The first binary code was the Braille alphabet.



- These days, most text information is stored in computers using a 7-bit code called the American Standard Code for Information Interchange (**ASCII**). There are $2^7 = 128$ ASCII codewords—more than enough to represent any character on a keyboard. Because the bits in a modern computer are generally grouped into **bytes** of 8 bits, one 7-bit ASCII codeword is usually assigned 1 byte of computer memory.

ASCII CODE: CHARACTER TO BINARY

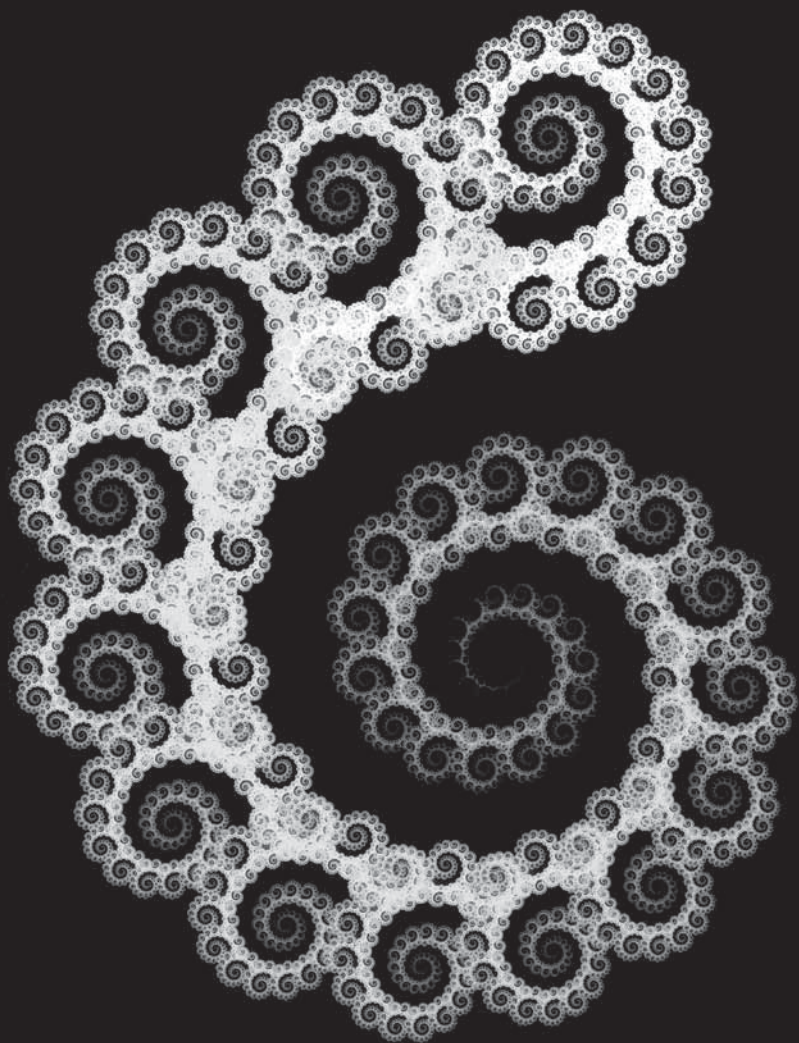
0 0011 0000	I 0100 1001	a 0110 0001	# 0111 0011
1 0011 0001	J 0100 1010	b 0110 0010	t 0111 0100
2 0011 0010	K 0100 1011	c 0110 0011	u 0111 0101
3 0011 0011	L 0100 1100	d 0110 0100	v 0111 0110
4 0011 0100	M 0100 1101	e 0110 0101	w 0111 0111
5 0011 0101	N 0100 1110	f 0110 0110	x 0111 1000
6 0011 0110	O 0100 1111	g 0110 0111	y 0111 1001
7 0011 0111	P 0101 0000	h 0110 1000	z 0111 1010
8 0011 1000	Q 0101 0001	i 0110 1001	. 0010 1110
9 0011 1001	R 0101 0010	j 0110 1010	, 0010 0111
A 0100 0001	S 0101 0011	k 0110 1011	: 0011 1010
B 0100 0010	T 0101 0100	l 0110 1100	; 0011 1011
C 0100 0011	U 0101 0101	m 0110 1101	? 0011 1111
D 0100 0100	V 0101 0110	n 0110 1110	! 0010 0001
E 0100 0101	W 0101 0111	o 0110 1111	' 0010 1100
F 0100 0110	X 0101 1000	p 0111 0000	" 0010 0010
G 0100 0111	Y 0101 1001	q 0111 0001	(0010 1000
H 0100 1000	Z 0101 1010	r 0111 0010) 0010 1001
		space 0010 0000	

The Two-Sided Nature of Information

- The use of computers introduces a new aspect of information. The bits stored in the computer's memory might simply be data—input, output, and so on. These are organized into identifiable blocks of memory, usually called *files*.

- But the bits might also represent computer **programs**, groups of instructions that tell the computer how to operate.
- When we use a program, we say that the computer “runs” or “executes” the instructions. But a stored program is just a file like any other; it can be moved, copied, changed, or erased. Thus, computer information has a two-sided nature: both data and program.
- ▶ One of the first people to understand this fact was the Hungarian-American mathematician John von Neumann. In the years following World War II, von Neumann was thinking about computers and robots—both the technical issues of how to build them and the basic principles on which they operate. One question von Neumann considered was this: Would it be possible to build a self-reproducing machine?
 - Imagine a robot that lives in a warehouse full of machine parts. The robot grabs components from the shelves and puts them together, eventually assembling an exact duplicate of itself. Von Neumann wondered how such a machine would work.
 - Obviously, a von Neumann robot would have to contain a complete set of instructions for building robots; in effect, it must contain a detailed blueprint of itself. But that raises a tricky question: Does the blueprint of the robot contain, as part of the blueprint, the blueprint itself?
- ▶ We can imagine a picture that contains a miniature copy of itself. We can construct a geometric shape, part of which is an exact copy of the whole (a *fractal*). But self-similar mathematical objects are always infinite in some way. Because they contain infinitely many points, they can have infinitely fine levels of detail. But that won't work for a von Neumann robot. Its memory is strictly finite.
- ▶ Von Neumann figured out that his machine had to work in a different way—a way that relies on the two-sided nature of information.

FRACTAL

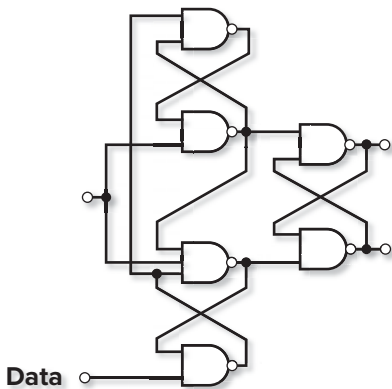


- The blueprint is a program that tells the robot how to construct its duplicate. But that program is also a kind of data file that can be copied and transmitted.
- The last step in the robot-building program might read, “Make a copy of the robot-building program in the new robot’s memory.” The self-reproducing program does not have to be infinite; it just has to be able to refer to itself.
- ▶ **Von Neumann machines** already exist in nature; they are called living things. All kinds of biological organisms produce offspring of the same basic design as themselves. Thus, von Neumann’s insight tells us something significant about how information works in living systems.
 - We now know that the genetic information in an organism is contained in its **DNA**; the sequence of chemical bases in a DNA molecule is a kind of blueprint for the organism.
 - Von Neumann’s thought experiment tells us that the information in DNA must be used in two quite different ways: (1) It must be expressed; that is, it must act as a program for the organism to build and operate itself. (2) It must be copied, like the blueprint of the von Neumann robot, during reproduction. And that is exactly what we find in nature.
- ▶ Notice the progression here: The idea of information begins with technology: communication, codes, information storage, computers, and robots. But the same ideas also apply to the natural world.

The Interplay of Ideas and Inventions

- ▶ In this course, we are interested in how the science of information has developed over time—the interplay of ideas and inventions. In the case of our current age of information, the idea and the invention arrived together.

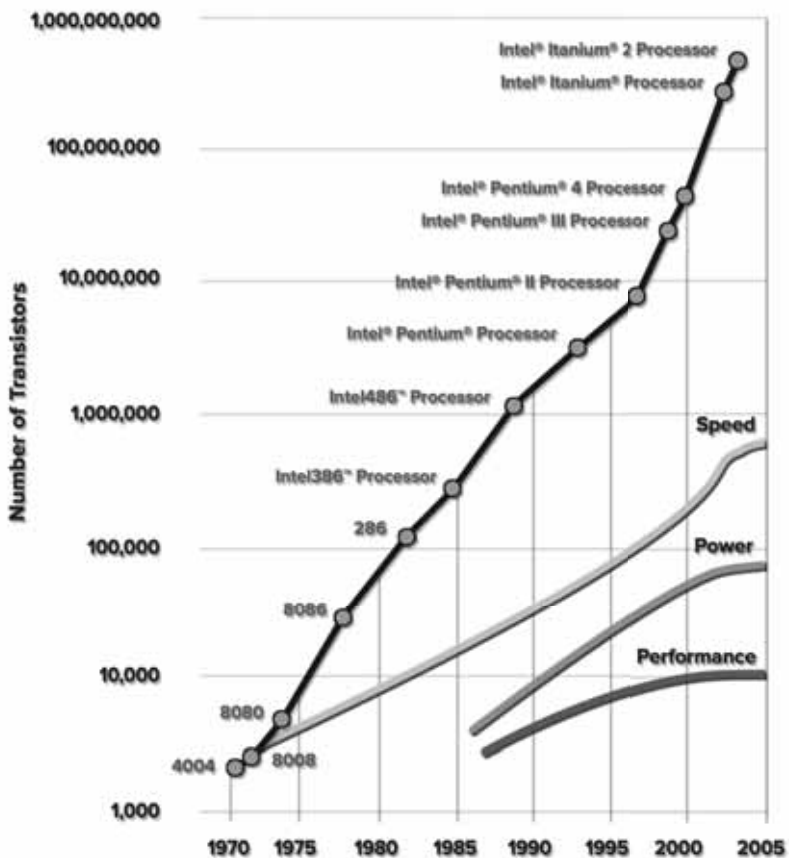
- The idea was Shannon's information theory, which taught us how to distill messages to bits. The invention was the **transistor**, developed just a few months before Shannon's paper was published.
- With about six transistors, a simple **flip-flop** circuit can be made. That's the basic memory module of a computer, where 1 bit of information can be written, stored indefinitely, and read back later.



FLIP-FLOP CIRCUIT

- Since the mid-20th century, the transistor has become the basis for more and more amazing technologies, especially those for communication and information processing. Those technologies have become almost unimaginably powerful and complex.
 - One way to chart this progress is the famous **Moore's law**, according to which the number of transistors on one integrated circuit has approximately doubled every two years. That rate of growth has continued, without interruption, for the last 50 years.
 - Almost every measure of information-processing power—speed, size, cost, energy consumption—has followed a similar exponential line of improvement. The result has been the creation of the technological world in which we live, the age of information.

MOORE'S LAW



TERMS

ASCII: The American Standard Code for Information Interchange, a widely used 7-bit code for text data first introduced in the early 1960s to represent any key on a standard English keyboard, including numerals, punctuation marks, and both uppercase and lowercase letters. Because most computers store ASCII symbols as bytes of 8 bits, the first bit is always a 0.

bit: The fundamental “atom” of information, the binary distinction between two alternatives (1/0, yes/no, on/off). Any type of information can be expressed as a sequence of bits.

byte: A group of 8 bits in a computer memory, generally the smallest block of information that can be addressed individually.

DNA: Deoxyribonucleic acid, the famous double-helix molecule that is the basic genetic information storage system for Earth life.

flip-flop: A combination of Boolean logic gates with a “feedback” from output to input, which can serve as a simple memory unit for 1 bit.

Moore’s law: The observation by computer engineer Gordon Moore that the power of information-processing devices increases exponentially over time. The exact law is variously stated; one form says that the number of transistors on a single processor chip roughly doubles every two years.

program: A data file that can also serve as a set of instructions for a computer.

transistor: A solid-state electrical component that allows one electrical signal to control another, invented in 1948 by John Bardeen, Walter Brattain, and William Shockley, all of Bell Labs. Basic logic gates can be built out of a few transistors.

von Neumann machine: A hypothetical robot able to assemble a copy of itself from a supply of basic components. John Von Neumann showed that it had to contain a set of instructions in its memory that was both expressed (in the building process) and copied as data (into the memory of the “offspring” machine).

READINGS

Golomb, et al., “Claude Elwood Shannon (1916–2001).”

Pierce, *An Introduction to Information Theory*, chapters 1–2.

Shannon and Weaver, *The Mathematical Theory of Communication*, “The Mathematical Theory of Communication” and Weaver’s nontechnical essay.

QUESTIONS

- 1 How many different characters can be represented in Unicode, which uses 16 bits per symbol?
- 2 Examine the files on your own computer. What different types of information are represented there in the common “currency” of bits? Do not count the system and software application files.
- 3 Investigate the history of the word *information*. Is it an old word or a recent coinage? Does the history of the word accord with its modern use?

Computation and Logic Gates

LECTURE

2

In Shannon's information theory, the essence of information is the distinction between different possible messages. The simplest possible distinction—the “atom” of information—is the bit, the binary distinction between 1 and 0, yes and no, or on and off. As Shannon pointed out, every sort of information—numbers, text, sound, images, and video—can be communicated by means of bits. In thinking about the power of bits, Shannon was following up an idea that he himself had introduced a decade earlier in his master's thesis at MIT. In it, he laid the basic groundwork for an amazing technological advance: the design of a computer out of electrical circuits, rather than mechanical parts.

The Development of Calculation Devices

- ▶ Since ancient times, people have sought ways to facilitate the painstaking work of numerical calculation. In the 19th century, one approach was to use tables of calculations that had been worked out in advance, including astronomical tables, navigational tables, tables of logarithmic and trigonometric functions, and so on.



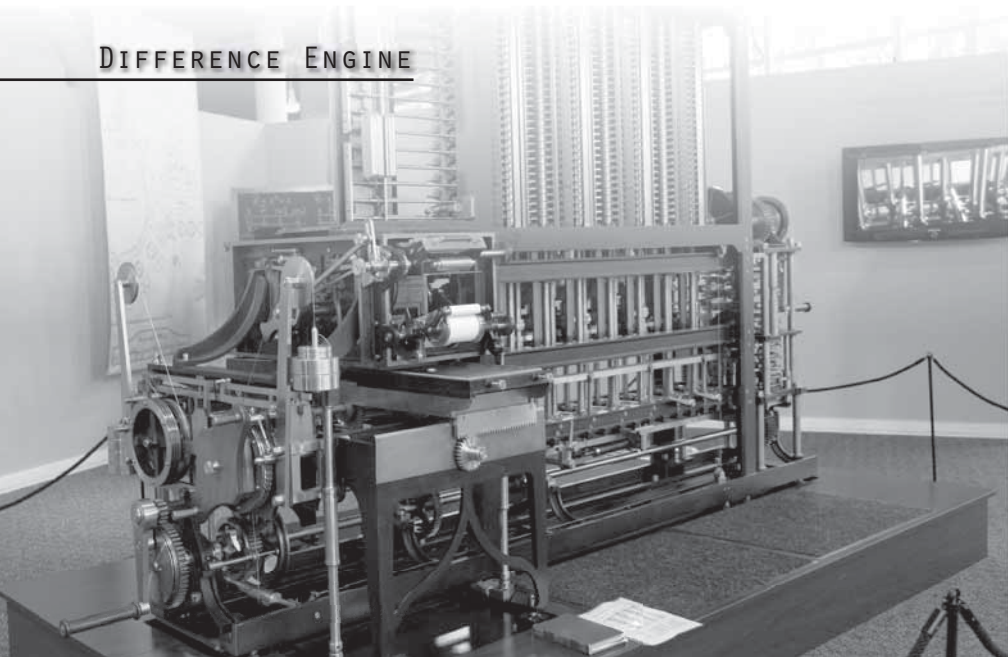
- In 1821, an English mathematician and scientist named Charles Babbage, unhappy with the system of tables, proposed a completely mechanical calculation machine—a device that could do calculations with perfect accuracy and print results without mistakes. Babbage's machine was called the **difference engine**.

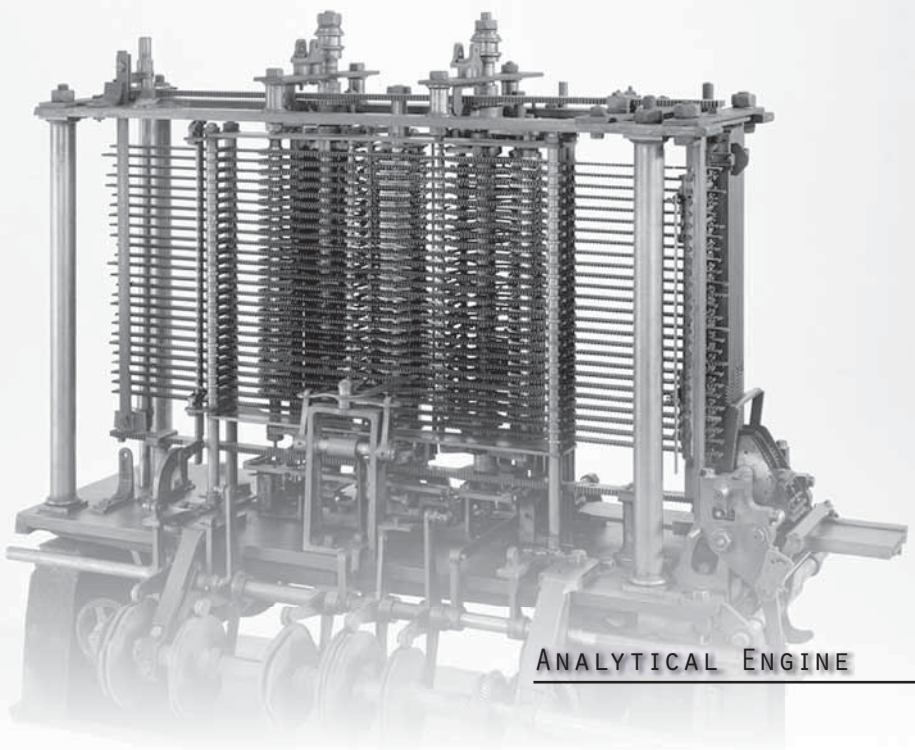


CHARLES BABBAGE

- Later, Babbage proposed a more ambitious design, the **analytical engine**. This would contain a memory unit called the *store* and a processing unit called the *mill*, and could be programmed to do any type of calculation. The program of the analytical engine would be represented by cards with holes punched in them. With the analytical engine, Babbage had invented the idea of a general-purpose computer.

DIFFERENCE ENGINE





ANALYTICAL ENGINE

- ▶ Babbage's dream remained just that during his lifetime and for long afterward. His most important ideas, such as the analytical engine, were largely forgotten until they were reinvented in different form a century later. But the difference engine, though it was never completed, inspired generations of mechanical calculating machines, including the ***differential analyzer***.
 - The name *differential analyzer* actually refers to several machines built during the late 19th and early 20th centuries, all of them designed to solve differential equations—the governing equations in many branches of physics and engineering.
 - These days, we would call the differential analyzer an analog computer. That is, it computed relationships among continuous variables, such as time, velocity, and position.

- ▶ In 1936, Claude Shannon was working as a technician on the differential analyzer at MIT and became fascinated, not with the mechanical parts of the device but with its electrical control system. Shannon realized that purely electrical components—switches and relays—could do calculations all by themselves. He addressed this problem in his master's thesis for a degree in engineering: how to make a computer out of electrical circuits.

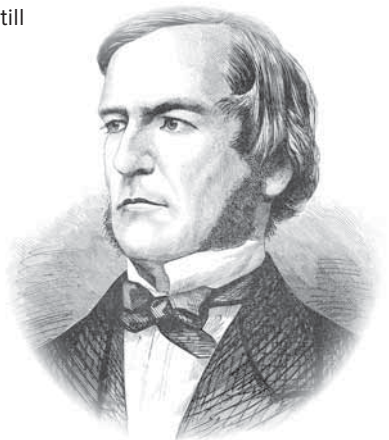
Understanding Electrical Circuits

- ▶ The simplest electrical circuit we can imagine consists of a battery, a switch, and a light bulb. We can understand what happens in this circuit in three different ways.
 - First, the circuit is basically a flow of electric charge. Electrons move in the wires. When the switch is closed, electric current goes through the wires around the circuit. Because electrons repel each other, a closed path (a circuit) is needed so that current can continue to flow.
 - The second way to view an electric circuit is as a flow of energy. The electrons leaving the battery have a higher energy than those returning. The battery adds energy to them. They also lose energy passing through the light bulb. The filament heats up and the bulb emits light. Thus, energy enters the circuit at the battery and leaves the circuit at the bulb. Energy flows from battery to bulb.
 - The third way to think about an electric circuit is as a flow of information. The bulb lights up only when the switch is closed and the current can flow freely, carrying energy from battery to bulb. The switch could be in one place and the bulb in another place, far away. The bulb would indicate whether the switch was closed. Thus, we could send a message by opening and closing the switch, and the light would flash off and on at the other end. In this view, information flows from switch to bulb.

- What kind of information are we talking about in the third view of an electric circuit? In the simple circuit we're considering, it is simply the binary distinction between on and off—1 bit of information. That's the aspect of the electric circuit that Shannon cared about when he pondered how to make an electrical computer.

Boolean Algebra

- In the late 1930s, information theory and the definition of a bit were still years in the future. Thus, Shannon turned to the work of an English mathematician named George Boole. Boole's achievement was to reduce logic itself to a kind of algebra. Logical relations and arguments take the form of algebraic operations and calculations. Shannon realized that this kind of *Boolean algebra* was the fundamental basis for any computation.



GEORGE BOOLE

- In Boolean algebra, the variables a , b , c , and so forth stand for logical statements. These variables can take just two possible logical values: true and false, which can be represented by 1 and 0. A Boolean variable, in other words, is a way of thinking about a bit.
- Shannon identified Boolean variables with 1-bit messages traveling on the wires of an electric circuit. What can we do with these messages? Boole had introduced several algebraic operations on them, and Shannon was able to show how each of these corresponded to a way of putting together a circuit.

- For instance, the simplest Boolean operation is negation, or NOT. If a is true, then “NOT a ” is false, and vice versa. This is represented by a symbol: NOT.
- In an electrical circuit, this would be represented by a kind of relay switch, a switch on one wire that is controlled by another wire. This can be connected so that the relay switch is on when the input from the control wire is off and vice versa. This is our first example of a *logic gate*.
- ▶ A logic gate stands for both a Boolean operation and a piece of electrical circuitry. Shannon’s idea was that these are really the same things. One or more inputs—1-bit messages—arrive from one side, and one or more 1-bit outputs emerge from the other side.
 - The next logic gate is the AND operation. Given two variables, a and b , then “ a AND b ” is true only when both a and b are true.
 - Another logic gate is the OR operation, which is a complement to AND. Here, “ a OR b ” is true if a is true, b is true, or both are true.
 - There is another natural definition of the OR operation, called the exclusive OR, or **XOR**. Here, “ a XOR b ” is true if either a is true or b is true, but not both.
- ▶ Shannon realized that any mathematical operation can be reduced to a basic logical pattern, which can, in turn, be translated into the layout of an electrical circuit. By using 1-bit messages in wires and connecting them to logic gates, any calculation can be performed just by electrons moving around. This simple idea laid the foundation for the modern electronic computer.

Working with Logic Gates

- ▶ We can get a sense of how this works by designing a circuit that can add a pair of 1-bit numbers together (a **half-adder circuit**) and one that can add three 1-bit numbers together (a **full-adder circuit**).
- ▶ We can also make other constructions: a NOT loop, in which a copy of the output of a NOT gate is fed back into the same gate as its input; a NOT NOT loop, in which two NOT gates are put in a row and the last output is fed back as input; a NAND (NOT AND) gate, that is, an AND gate followed by a NOT gate; and a NAND NAND gate, in which the output of each gate is fed back to the other gate as input. This last is a flip-flop circuit, a 1-bit memory unit.
- ▶ Note that each of these circuits is a combination of logic gates, connected by wires carrying 1-bit messages. The full adder is a communication network linking 17 logic gates. The flip-flop memory circuit uses just two NAND gates and a feedback network that lets the gates talk to themselves. These simple pieces can be linked to create modules with increasingly complicated functions, and the modules can then be used in larger circuits.
- ▶ These same ideas, very far extended, are the basis for the design of modern computer processors.
 - A single processing chip might contain hundreds of millions of logic gates, each one consisting of a few microscopic transistors, connected by wires only a few hundred atoms wide, exchanging 1-bit messages billions of times per second.
 - That is an extraordinary level of complexity, but it is all built out of simple **Boolean logic** gates. That's what Shannon understood as he wrote his master's thesis.

- The bit—the distinction between 1 and 0—is the fundamental “atom” of information. Every sort of information can be represented by bits. And in the same way, 1-bit messages and simple Boolean logic gates are the fundamental “atoms” of computation. Every sort of computation can be constructed from them.

TERMS

analytical engine: The general-purpose mechanical computer whose design was sketched, but never completed, by Charles Babbage in the 19th century, anticipating most of the key ideas of modern computers. The computer could be programmed using punched cards similar to those that controlled the Jacquard loom, with a “mill” for performing computations and a “store” for working memory.

Boolean logic: The algebraic logic devised by George Boole in the 19th century, used as the basis for the design of fully electronic computers in Claude Shannon’s 1937 master’s thesis. Shannon showed that any mathematical calculation could be reduced to a complex network of Boolean operations and that these could be implemented via electrical circuits.

difference engine: The first mechanical computer devised by Charles Babbage in order to create and print mathematical tables without human error.

differential analyzer: One of several related analog computers of the early 20th century. Each used special gear systems to connect the rotations of different shafts, the mechanical movement of each shaft representing a continuously changing variable, together representing the solution to a system of differential equations, such as the equations governing ballistic motion.

full adder: A combination of Boolean logic gates that computes the sum of three 1-bit inputs and produces the answer as a 2-bit binary number. Because the full adder can include the results of a “carry” operation, a cascade of these devices can accomplish the addition of two binary numbers of any specified length.

half adder: A combination of an AND gate and an XOR gate that computes the sum of two 1-bit inputs and produces the answer as a 2-bit binary number. The half-adder is a component part of a full adder.

XOR: The “exclusive OR” operation in Boolean logic. When both input bits agree (00 or 11), then the output is 0; when they disagree (01 or 10), the output is 1.

READINGS

Gleick, *The Information*, chapters 4–6.

The Logic Lab, www.neuroproductions.be/logic-lab/.

QUESTIONS

- 1 One of the mathematical problems analyzed by Shannon in his 1936 master’s thesis was the problem of constructing a table of all prime numbers up to 100,000,000. Shannon wrote: “J. P. Kulik spent 20 years in constructing a table of primes up to 100,000,000 and when finished it was found to contain so many errors that it was not worth publishing.” What would Charles Babbage have thought of this?
- 2 Some Boolean logic gates can be created from others. Show how an OR gate can be built out of a combination of AND and NOT gates.

Measuring Information

We have seen how any mathematical calculation can be performed by a complex network of 1-bit messages and simple logic gates. With this idea, Claude Shannon laid the conceptual foundations for modern electronic computers. In this lecture, we will begin to dig into information theory itself, as Shannon created it a decade later. We begin with the questions: How do we measure information, and how do we tell whether a message involves a small amount or a large amount of information?

Thinking about Information Measurement

- ▶ Let's imagine a source of information—something that produces messages. The messages might be speech, letters of the alphabet, electrical signals, or the daily high temperature.
- ▶ The number of possible different messages that might be produced (M) depends on the source. This number is significant because it does not depend on the choice of code (say, teletype versus ASCII). However, for whatever code we choose, we will need at least M different codewords.
- ▶ According to Shannon, information is really about the distinction between possible messages. Thus, it makes sense to imagine that the amount of information depends on M , the number of possibilities. If M is small—perhaps only 2 for a 1-bit message—then the amount of information in the message is small. If M is huge, then we gain a great deal of information when we receive the message.
- ▶ It's tempting to adopt M itself as our measure of information, but doing so doesn't work out as we would like. Consider two messages from the same source. How many possible pairs of messages are there?

- The first message has M possibilities, and for each of those, the second message also has M possibilities. Thus, the two messages have $M \times M = M^2$ joint possibilities. The numbers multiply.
- But our intuition says that two messages should have just two times as much information as one message, not the square of the information.
- To appease that intuitive idea, we need to take the number of possibilities, M , which is multiplicative for more than one message, and turn it into something additive. The right trick to do that is to use logarithms.

Understanding Logarithms

- Logarithms turn multiplication and division into the easier operations of addition and subtraction.
 - We settle on a number B (greater than 1), which is called the *base*. We can raise B to any power: positive, zero, negative, or fractional. A related fact is that any positive number can be written as some power of B . That is, given a number z , we can always say that $z = B^x$ for some exponent x .
 - In a nutshell, a logarithm is an exponent. For a base B , there is a function \log_B . And the two equations $z = B^x$ and $x = \log_B(z)$ mean the same thing.
 - Thus, choosing $B = 2$, we have: $\log_2(1) = 0$, because $2^0 = 1$; $\log_2(2) = 1$, because $2^1 = 2$; $\log_2(4) = 2$, because $2^2 = 4$; $\log_2(8) = 3$, because $2^3 = 8$; $\log_2(3) = 1.585$, because $2^{1.585} = 3$; and so on.
- Because a logarithm is just an exponent, all the basic rules of exponents are also rules for logarithms. For example:
 - The logarithm of the product of two numbers is the sum of their logarithms: $\log_B(x \times y) = \log_B(x) + \log_B(y)$.

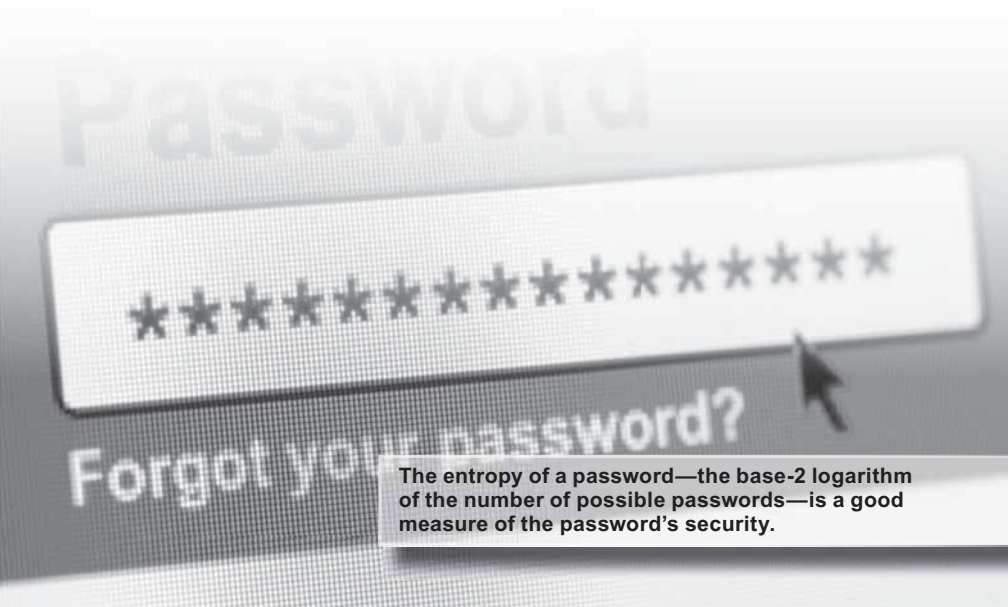
- The logarithm of a quotient of two numbers is the difference of their logarithms: $\log_B(x/y) = \log_B(x) - \log_B(y)$.
 - The logarithm rule for powers: $\log_B(x^n) = n \times \log_B(x)$.
 - The logarithm rule for reciprocals: $\log_B(1/x) = -\log_B(x)$.
 - $\log_B(1) = 0$, no matter what base B we choose.
 - And as a general fact, if $x > y$, then $\log_B(x) > \log_B(y)$.
- What base number B should we choose? In information theory, we use the base-2 logarithm, \log_2 .
- To calculate \log_2 , note that the natural logarithm of 2 is $\ln(2) = 0.693$. Then, it turns out that $\log_2(x) = \ln(x)/\ln(2)$, or $\ln(x)/0.693$. For large numbers, a quick rule of thumb to estimate the base-2 logarithm is to count the digits in the number.
 - For example, 5,000,000,000 has 10 digits. According to the rule of thumb, we multiply the number of digits by 10/3: $10 \times 10/3 = 100/3$, or 33.3. (The base-2 logarithm of 5 billion is actually closer to 32.2.)

Understanding Entropy

- As we said, our information source produces M possible messages. According to Shannon, the amount of information contained in a message from the source is measured by the **entropy**, which is defined as: $\text{entropy } H = \log_2(M)$.
- This definition inspires a number of questions. First, why use the logarithm? For the answer, remember what happens if we have a pair of messages from the source.
- The total number of message pairs is M^2 . Thus, the entropy of a pair is $\log_2(M^2) = 2 \times \log_2(M) = 2 \times H$, where H is the entropy of one message. In other words, the entropy of the pair is just twice

as much. The logarithm in the entropy has made our measure of information additive, as we wanted.

- What happens when $M = 1$? The entropy of such a message is zero, because $\log_2(1) = 0$. That makes sense: If there is only one possible message, then you already know what the message is, even before you receive it.
- ▶ Why should we use base-2 logarithms? This is an arbitrary choice, but the base-2 logarithm makes things work out neatly in information theory.
 - If we have a 1-bit message source, $M = 2$; thus, $H = \log_2(2) = 1$.
 - In effect, using base-2 logarithms means our entropy is measured in bits.
- ▶ Why do we call this the *entropy*? From the outset, Shannon knew that there was an analogy between his measure of information and an important concept in thermodynamics. In that field, entropy helps determine whether a particular energy transformation is possible or not, and it can be calculated by taking the logarithm of a certain number.



- What does *entropy* mean? We can interpret H in two ways: (1) the amount of information we gain when we receive a message, that is, a measure of the content of the message, or (2) the amount of information we lack before we receive a message, that is, a measure of our uncertainty about the message.

Entropy and Information Theory

- Entropy also tells us something important about the codes we can use to communicate messages. Again, suppose that our source generates M different possible messages.
 - To take a concrete example, let $M = 5$; the message is one of the letters A through E. We want to create a code that represents the output of this source as a codeword made up of binary digits. How many bits do we need?
 - The code must have as many codewords as there are messages, so that no two messages have the same codeword. If we use codewords that are n bits long, then there are 2^n codewords available, from 0000 to 1111. Thus, 2^n must be at least as great as M .
 - But that relation about exponents can be recast as a relation about logarithms. That is, n is at least as great as $\log_2(M)$, which is the entropy, H .
- The number of bits we need to represent a message from the source must be at least as great as the entropy of the source.
 - We could use more bits than H to encode the message—there is no limit to how inefficient our code might be—but we cannot get by with less than H bits per message.
 - For the source that produces letters A through E, the entropy is $\log_2(5) = 2.32$. Therefore, we need at least 3 bits to represent the output of the source.

- ▶ Suppose our source produces a series of messages, one after the other, with each message independent of the last one.
 - We could then group the messages together and make a code, not for individual messages but for successive pairs of messages. There are 25 possible pairs: AA, AB, and so on, all the way to EE. The entropy of the pair is 4.64—just twice the entropy of one message. That means that we can code a pair of messages with a 5-bit codeword.
 - We use 5 bits to represent two messages; thus, in some sense, we use $5/2 = 2.5$ bits per message.
 - This approach is called *block coding*. Instead of coding messages one by one, we create codewords for entire blocks of several messages, which enables us to be more efficient, using only 2.5 bits per message instead of 3. That savings should allow us to communicate more quickly or store more information in a given space.
- ▶ Using 2.5 bits per message is still greater than the fundamental entropy limit, which is 2.32 bits per message, but we have gotten closer to that limit. How close can we get? Let's try coding triplets of messages.
 - There are $5^3 = 125$ possible triplets, from AAA up to EEE. The entropy of a triplet is three times the single-message entropy, or 6.96 bits. That means that a code of 7 bits could accurately represent the triplets.
 - And using 7 bits for three messages means $7/3 = 2.333$ bits per message, which is quite close to the entropy limit of 2.32 bits per message.

Shannon's Conclusions

- ▶ Shannon drew a general conclusion: The entropy, H , of the message source yields a lower limit to the size of the binary codewords in any code that can faithfully represent the message. By coding blocks together and choosing an efficient code, we can approach this limit as closely as we like on a per-message basis.

- ▶ Therefore, the entropy, H , of a source equals the minimum number of bits required to represent a message from the source. Because that's a somewhat informal statement, we need to be careful about its interpretation. H is always a lower limit to the number of bits we need, but we can approach that limit as closely as we wish by taking the wholesale approach and coding many messages together.
- ▶ Entropy, then, is what we have been seeking:
 - It is a measure of the information in a message that does not depend on how the message is represented.
 - It's an additive measure. The entropy of a pair of messages is just the sum of the entropies of the individual messages.
 - And it is related to the length of the message, specifically, to the number of bits the message requires if we use an efficient code.
- ▶ Here is another way of thinking about the meaning of entropy: Each bit in a codeword—each 1 or 0—can be thought of as the answer to a yes/no question. The entropy tells us how many yes/no questions we would need to ask before we could determine the message. The code is just a convenient way of organizing the questions.

TERM

entropy: The fundamental measure of information. There are several closely-related definitions. The *Hartley-Shannon entropy* of a message source is the base-2 logarithm of the number of possible messages; it applies when all of the possible messages are equally likely. When the messages differ in probability, then the *Shannon entropy* is used. In both cases, Shannon's first fundamental theorem tells us that the entropy equals the minimum average number of bits needed to represent the message in a binary code. *Thermodynamic entropy*, as discovered by Rudolf Clausius and Ludwig Boltzmann in the 19th century, is simply proportional to the information entropy of the unknown microstate of a macroscopic system. Some modern

departures from Shannon's information theory have their own revised definitions of entropy. The *algorithmic entropy* of Andrey Kolmogorov and Gregory Chaitin is the length of the shortest computer program that can produce a particular binary string as its output. In quantum mechanics, the *von Neumann entropy* (formulated by John von Neumann) takes into account that “nearly” quantum states are not fully distinguishable.

READINGS

Gleick, *The Information*, chapters 7–8.

Pierce, *An Introduction to Information Theory*, chapters 3–4.

QUESTIONS

- 1 Suppose we have an information source that produces three possible messages, for example, by lighting up colored bulbs that might be red, green, or blue (R, G, or B). How long must the codewords be in a binary code that represents sequences of colored lights of length 1, 2, 3, 4, or 5? How many bits are used per light signal in each case?
- 2 In the state of Ohio, new license plates typically have three letters followed by four numerical digits: XYZ 1234. How many bits of car-identification information are found on a license plate?
- 3 Suppose Alice is thinking of a number between 1 and 100 (inclusive), but Bob does not know the number. How many bits of information is Bob missing? Now he asks Alice whether her number is even. When he learns the answer, how many bits will he still be missing? How many more yes/no questions will he need to learn the number?
- 4 You are playing 20 questions, and you know that the unknown “target” is a word from the *Oxford English Dictionary*. Suggest a strategy for asking questions to find the word using as few questions as possible.

Entropy and the Average Surprise

Our everyday intuition says that we measure information by looking at the length of a message: 5 bits, 1 kilobit, and so on. But Shannon's information theory starts with something more fundamental: How surprising is the message? How informative? A long message might not be very informative, while a short message might be quite informative. The question then becomes: How do we measure the surprise of a message?

Messages and Probability

- ▶ So far, we have considered only a simple model for an information source. The source produces one of M different possible messages; each possible message is equally likely. Furthermore, any two successive messages are completely independent of each other.
 - But this simple model of an information source does not closely resemble a real source of messages. We need an idea of an information source that includes some of the patterns and regularities of actual messages.
 - Claude Shannon's approach to this problem was to use the concept of probability.
- ▶ The theory of probability was introduced in the 17th century as a way of reasoning in the presence of uncertainty. We assign each possible event, x , with a measure of its likelihood, called its *probability*, $p(x)$.
- ▶ The first rule of probability is that the probability $p(x)$ is always between 0 and 1: For any x , $0 \leq p(x) \leq 1$. If x is an impossible event, then $p(x) = 0$. If x is absolutely certain to occur, then $p(x) = 1$. In other situations, $p(x)$ is somewhere in between—closer to 0 if x is less likely and closer to 1 if x is more likely.

- ▶ The second general rule about probability is that the probabilities add up to 1. Stated more carefully, for any random variable X with values x : $\sum_x p(x) = 1$.
 - Here, the Greek letter Σ stands for “sum,” and it tells us to add up $p(x)$ over all the possible values of x . If the possible values are $\{a,b,c\}$, then the sum just means $p(a) + p(b) + p(c)$.
 - In our text-generating machine, the random variable includes all the symbols that the machine might produce: the 26 letters, plus the space. Each of those letters will be assigned its own probability. (An assignment of probabilities is called a *probability distribution*.)
- ▶ How do we decide what probability distribution to choose to make a realistic text-generating machine? Our choice hinges on how we interpret probability numbers.
 - Imagine a great many similar situations, such as flipping a coin many times. Those situations represent different *trials* of the same coin-flipping experiment. In some trials, the event x occurs—the coin comes up heads—but in others, it does not.
 - Suppose N is the total number of trials and $N(x)$ is the number of times x occurs. In the long run, then, we expect: $p(x) = N(x)/N$.
 - The left-hand side is called the *statistical frequency* of x , the fraction of the trials in which x occurs.
 - In the long run, our claim is that probability should be the same thing as statistical frequency. The coin comes up heads half the time.
 - This is a useful intuition, but it’s also tricky. This equation is not literally true, especially if the number of trials is small. If we flip a coin 10 times, we might get only three heads, even if it is a perfect 50-50 coin. The best we can say is that probabilities and frequencies are very likely to be nearly equal if we make many trials of our experiment.

- We will base the letter probabilities for our text-generating machine on the letter frequencies in a reference text, in this case, *The Return of Sherlock Holmes* by Sir Arthur Conan Doyle. This book contains about 580,000 characters, making up around 112,000 words. In this text, the most common character is the space (occurring 19.3% of the time), followed by the letters E (10.0%), T (7.3%), A (6.6%), and so on. This knowledge allows us to assign reasonable letter probabilities for our text-generating machine.

Measuring Message Surprise

- We now have an information source whose messages—the letters—are not equally likely. How much information is being produced by the source? To answer that question, let's think about a simpler example, an information source that produces only two possible messages, such as a fire alarm.

The fact that the probability of “no fire” is close to 1 and the probability of “fire” is close to 0 tells us that we need a measure of surprise to measure the information content of a message.



- ▶ At any given moment, the fire alarm sends a message that says “no fire” (the alarm is silent) or “fire” (the alarm is sounding). This seems at first like a simple binary source of information. At each moment, we learn 1 bit of information.
- ▶ But that conclusion feels wrong. To us, it seems as if we learn more when the alarm goes off than we do when the alarm stays silent.
 - At any given moment, it is overwhelmingly more likely that there is no fire; $p(\text{no fire})$ is close to 1, and $p(\text{fire})$ is close to 0. Thus, we more or less expect the alarm to be silent.
 - When the alarm sounds, the message has a larger effect on our thinking. To measure the information content of a message, therefore, we need a measure of its **surprise**.
- ▶ Let’s designate the surprise of message x to be $s(x)$. This depends on the likelihood of the message; a less probable message has more surprise to it. In information theory, the measure of surprise is based on the logarithm

$$s(x) = \log_2 \left(\frac{1}{p(x)} \right).$$

The surprise of x is the logarithm of the reciprocal of the probability of x . This definition has all the characteristics we want.

- If $p(x) = 1$, then $s(x) = \log_2(1) = 0$. A message that is certain has zero surprise.
- When the probability is less than 1, its reciprocal is greater than 1, and the surprise is a positive number.
- If $p(x)$ gets very close to zero, $\frac{1}{p(x)}$ becomes very large, and the surprise $s(x)$ gets large, as well.
- If $p(x) = 0$, the surprise blows up mathematically. A truly impossible event would be infinitely surprising. But because an impossible event can never happen, we are never actually infinitely surprised.

- Recall one of our old-style information sources, in which the M possible messages are all equally likely. That means that each message must have probability $1/M$, so that M of them will add up to 1. The surprise of such a message is

$$s(x) = \log_2 \frac{(1)}{1/M} = \log_2(M).$$

That is exactly the Hartley-Shannon entropy, which measures the amount of information that the message source produces. (Ralph Hartley was an engineer whose work was an important precursor to Shannon's.)

- Surprise is a measure of the information in a message; it's measured in bits. If all messages were equally likely, they would all have the same surprise. But if the messages have different probabilities, the less likely messages convey more information.
 - A model of English in which all letters are equally likely is called a *0th-order letter-by-letter model*. A text-generating machine based on that model isn't very realistic.
 - A model that takes into account the different probabilities for the different letters is a *1st-order letter-by-letter model*. But the output of a machine that uses this model is only a little better because the letters in English text are not independent of each other. Each letter tells us something about the letters around it.
 - By studying pairs of letters in our reference text, we can match not only the probabilities for the individual letters but also the conditional probabilities for the next letter given the last letter. This gives us a *2nd-order letter-by-letter model* of English.
 - In a *3rd-order letter-by-letter model* of English, the next letter depends on the last two letters. If we make a text-generating machine like this, the output seems more like realistic English text.

Generating Word Messages

- ▶ Up to now, we have been thinking about English as a series of letters, one by one. Suppose, instead, that we think about English as a series of words. Our text-generating machine will produce words one by one.
- ▶ When we analyze *The Return of Sherlock Holmes*, we count slightly more than 8,000 different words. We also note that some words are much more common than others. The top 10 most common words are: *the, I, and, of, a, to, that, in, was, it*.
- ▶ In 1935, the Harvard philologist George Kingsley Zipf observed that if we count the number of times each word appears in a given book, then arrange that list by the rank of the word—first most common, second most common, and so on—a mysterious pattern emerges: Roughly speaking, the number of occurrences of a word is inversely proportional to its rank in the list.
 - That means that the second most common word occurs about half as often as the most common word, the third most common word occurs about one-third as often, and so on. Word 100 on the list occurs about twice as often as word 200.
 - This empirical fact is called **Zipf's law**, and variations of it apply to other languages and to a surprising range of other sorts of data.
 - Zipf's law works fairly well for *The Return of Sherlock Holmes*. For example, the word *crime* is 218 on the list and occurs 60 times. Twice as far down in the list, at 436, is the word *master*, which occurs 28 times, about half as often.
- ▶ Using these data, we can construct a *1st-order word-by-word model* of English, a text-generating machine that produces entire words with the observed frequencies. We can also create *2nd-order* and *3rd-order word-by-word models* of English.
 - Each of our successive text-generating machines gives a recognizably better approximation or simulation of a real English text.

- Each of our models captures more of the patterns and regularities of actual language.
- ▶ The information sources considered in Shannon's theory generate messages according to probabilistic rules. That isn't exactly the way that you and I would produce English text, but as we've seen, a probabilistic model can do a fairly good job, especially if we are willing to consider 2nd- and 3rd-order versions that take correlations into account.

Message Surprise and Entropy

- ▶ Let's consider one of Shannon's information sources. There is a set, X , of possible messages containing M different alternatives. Message x occurs with probability $p(x)$. To keep things simple, we will still assume that successive messages are independent of each other. That is, we have a 1st-order message source.
- ▶ Previously, we said that the amount of information produced by the source was the Hartley-Shannon entropy, which is $\log_2(M)$. That measure of information was connected to everything from password security to the number of bits we need in a binary code.
- ▶ Now, the different messages have different probabilities, and the amount of information in each one is the surprise, the logarithm of the reciprocal of the probability. That's different for different messages. Yet we still need a measure of entropy, the overall amount of information in the source.
- ▶ How is the surprise of the message related to entropy of the source? Entropy is the average surprise. Shannon's formula for this is:

$$H(x) = \sum_x p(x) \log_2 \left(\frac{1}{p(x)} \right).$$

- ▶ What does this formula tell us about English text?

- If we take a letter-by-letter view, the old Hartley-Shannon entropy gives 4.76 bits per character. If we take into account the letter probabilities that we estimated from *The Return of Sherlock Holmes*, we find a lower value—around 4.07 bits per character.
 - If we take a word-by-word view of English, taking into account the different probabilities of different words, we end up with 9.25 bits per word—less than 10 bits.
 - The average English word is about five letters long. This suggests that the real entropy of English is less than 2 bits per letter.
- What does that “2 bits per letter” actually mean? Remember, we previously found a connection between entropy and coding.
- Entropy is the answer to the practical question of how many binary digits we needed in our codewords to represent the messages.
 - Now, Shannon has given us a new, more general definition of entropy: Entropy is the average surprise of a message. With unequal probabilities, that entropy is less.
 - Does that mean that we can use fewer binary digits in our codewords? Could we represent English with a code that uses only 2 bits per letter? Is there still a connection between entropy and coding? We'll explore these questions in the next lecture.

TERMS

surprise: The novelty of a random event, measured in bits, equal to the logarithm of the reciprocal of its probability. If event A is half as likely as event B, its surprise is 1 bit greater. The Shannon entropy of an information source is the average surprise of its messages.

Zipf's law: An empirical fact about language first noted by the linguist George Kingsley Zipf in 1935. Simply stated, in any long sample of English text, the N^{th} most common word is about N times less likely than the most common word. Similar regularities occur in other languages.

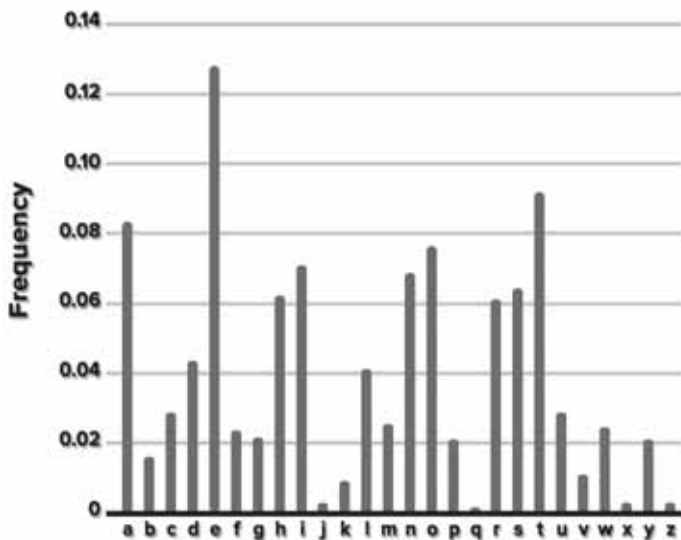
READINGS

The Dasher Project, www.inference.phy.cam.ac.uk/dasher/.

Pierce, *An Introduction to Information Theory*, chapter 5.

QUESTIONS

- 1 Look at the chart of the letter frequencies of English below. How many of the least-frequent letters would you have to combine to have about the same total probability as the most frequent letter, E? What is the total probability of the 12 most common letters (ETAOIN SHRDLU), according to the printer's rule?



- 2** Below is the table of weather probabilities from the lecture. Use it to find the surprise for each of the four possibilities. What is the most surprising weather? Also find the entropies $H(S)$, $H(T)$, and $H(S,T)$ using the Shannon formula.

		T	
		p	
S	sunny	0.4	0.2
	cloudy	0.1	0.3

- 3** It can be interesting to compute the surprise value for various familiar but rare events. Each year, about 50 people in the United States (out of 300 million) are victims of lightning strikes. Calculate the surprise for a given person to be struck and killed tomorrow. Compare it to the surprise for flipping a coin and getting 40 heads in a row.

Data Compression and Prefix-Free Codes

After 50 years of Moore's law, with its exponential improvement in computer and information technology, it has become absurdly inexpensive to store information. In fact, the cost of data storage is so cheap that it's measured in a new unit of money, the nanocent. One bit of data storage costs half a nanocent. Despite this affordability, it is still important to find codes that pack as much information into as few bits as possible, in part to enable us to transfer information from place to place. For this, we turn to the topic of data compression.

Information Inequality

- ▶ Shannon's theory is really about tasks and resources. We are given a task: to convey messages from one place to another or to store them for later retrieval. We also have some resources: binary digits, transmitted or stored as electrical signals; holes in paper tape; or tiny magnetic domains in a computer disk. The basic question that Shannon's theory aims to answer is this: What resources are required to accomplish the task?
- ▶ In the last lecture, we learned to measure information by surprise. For an information source X that produces message x with probability $p(x)$, the surprise, $s(x)$, is:

$$s(x) = \log_2 \left(\frac{1}{p(x)} \right).$$

This is the amount of information in the message, measured in bits. An unlikely message carries more information than a common one. According to Shannon, the entropy of the source X , called $H(X)$, is the average surprise of the messages in X .

- Let's return to a simple example to illustrate this idea. Our source has three possible messages: $\{a,b,c\}$. Message a has probability $1/2$, while b and c each have probability $1/4$. Thus, the surprise of a is 1 bit, and the surprises of b and c are each 2 bits. The average surprise, the entropy, is 1.5 bits.
 - Suppose that you know the actual probabilities, $p(x)$, for messages from the information source X . But another message recipient has some other, possibly mistaken probabilities in mind, designated $q(x)$.
 - This other recipient believes that all three messages are equally likely, so that $q(x) = 1/3$. Thus, each message for this recipient has a surprise of $\log_2(3) = 1.585$ bits.
 - You and the other recipient start receiving messages from the source. Sometimes, the other recipient is more surprised than you are. When a occurs, your surprise is 1 bit, and the other recipient's is 1.585 bits. Sometimes, the other recipient is less surprised. When b or c occurs, your surprise is 2 bits, but the recipient's is still 1.585 bits. On average, the other recipient is more surprised than you are.
 - In this example, the other recipient's surprise is always 1.585 bits, while your average surprise is only 1.5 bits. This is a general fact for any message source. The surprise of two recipients for a given message x is $\log_2\left(\frac{1}{q(x)}\right)$ and $\log_2\left(\frac{1}{p(x)}\right)$, respectively.
 - When we calculate the average surprise, though, we must take the average using the correct probabilities, that is, $p(x)$, not $q(x)$. Your average surprise is the entropy $H(X)$. We can now write a mathematical inequality:

$$\sum_x p(x) \log_2\left(\frac{1}{q(x)}\right) \geq H(X).$$

- The recipient with the wrong probabilities is always more surprised, on average. This mathematical fact shows up all through information theory, so much so that it is called the *information inequality*.

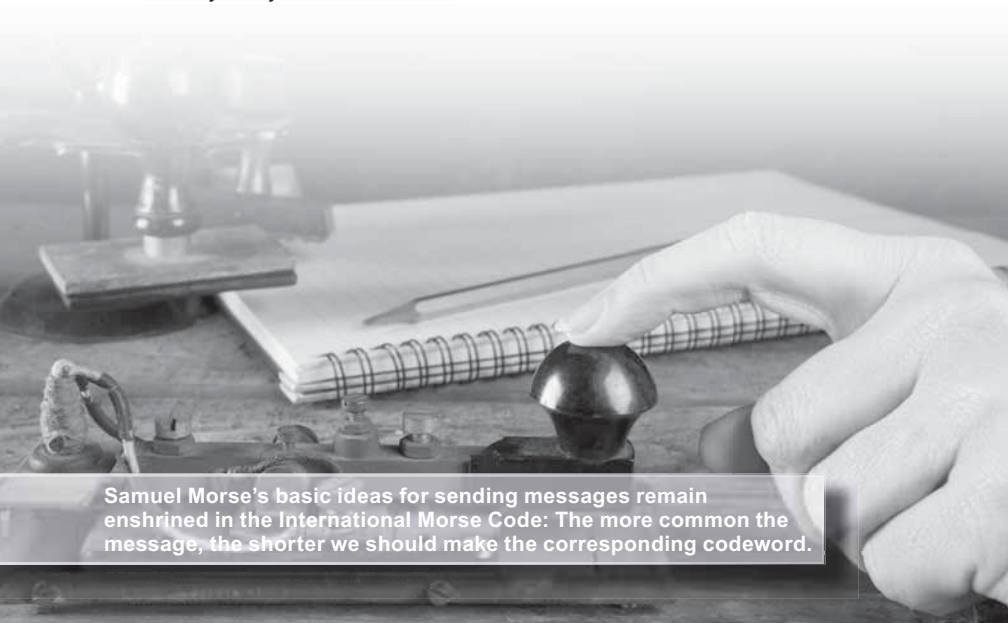
Morse Code

- If surprise is our measure of information, some messages carry more information than others. Does that mean that they require more bits to represent them? We have not yet discussed codes in which different messages are represented by different numbers of bits. This important idea goes back more than 100 years before Shannon, to the invention of the telegraph in the 1830s by Samuel F. B. Morse.



SAMUEL F. B. MORSE

- Morse was not the first person to dream of sending messages by electricity, but he succeeded where others failed because he kept it simple. A single pair of wires connected the sender and receiver. The signal was simply the on/off state of the input key. But how could a meaningful message be conveyed by “on-off-on-off”?



Samuel Morse's basic ideas for sending messages remain enshrined in the International Morse Code: The more common the message, the shorter we should make the corresponding codeword.

- ▶ Morse's idea was to represent each letter by a series of electrical pulses. These pulses came in two varieties, a short one (*dot*) and a longer one (*dash*). In terms of electrical signals, the dot was “on-off” and the dash was “on-on-on-off.” There were longer pauses between letters and between words.
- ▶ Morse used shorter patterns—shorter codewords—for the more common letters. In what came to be called **Morse code**, E is a single dot, T is a single dash, A is a dot followed by a dash, and so on. The code was designed so that messages would be conveyed in the shortest possible time.

MORSE CODE

A	• —	N	— •	1	• — — — —
B	— • • •	O	— — —	2	• — — — —
C	— • — •	P	• — — •	3	• • — — —
D	— • •	Q	— — — •	4	• • • — —
E	•	R	• — • •	5	• • • • —
F	• • — •	S	• • •	6	— • • • •
G	— — — •	T	—	7	— — — • •
H	• • • •	U	• • —	8	— — — • • •
I	• •	V	• • • —	9	— — — — •
J	• — — — —	W	• — — —	0	— — — — —
K	— • — —	X	— • • —		
L	• — • •	Y	— — • — —		
M	— —	Z	— — • •		

Efficiency in Messages

- ▶ Morse demonstrated that we can communicate with the greatest efficiency by using shorter codewords for common messages (or letters). But how efficient can we be?
- ▶ Let's consider a code built out of bits in which there are two possible codewords of length 1 (0 and 1); four possible codewords of length 2 (00, 01, 10, and 11); eight possible codewords of length 3; and so on.

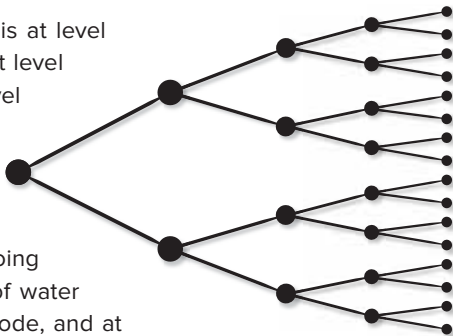
We can use any of these in our code, and we allow different lengths for different codewords.

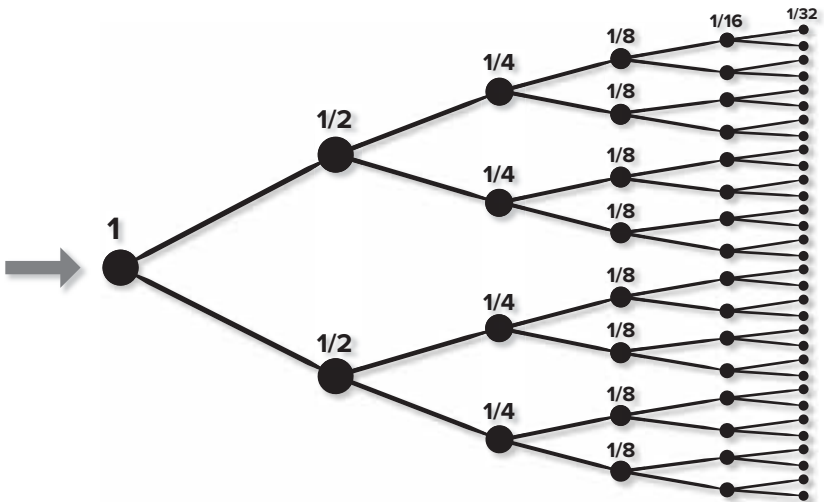
- ▶ Recall our three-message information source with unequal probabilities: $p(a)$ is $1/2$, while both $p(b)$ and $p(c)$ are $1/4$. Let's try the following code: a is 0, b is 1, c is 00. Here, each message has its own codeword, and the more common message has one of the shorter codewords, but there is a serious problem with this code.
 - Suppose we receive a series of messages in this code. We get a stream of bits, perhaps starting 0011100010.... Our job is to decode this in the $\{a,b,c\}$ alphabet.
 - The first bits are 001, but there's no way to tell whether that means aab (0-0-1) or cb (00-1). Both aab and cb are represented by the same binary digits, 001.
 - If we use this code for a series of messages, it fails the first principle of codes: Different messages cannot be represented in exactly the same way. We never ran into this problem before, because our previous codewords all had the same length. If that length were 3 bits, then we'd take the first 3 bits for the first codeword, the next 3 for the next codeword, and so on. We knew how to divide up a series of bits into individual codewords, but now that the codewords might have different lengths, we don't know how to divide them.
- ▶ We can put extra spaces between the codewords, but then, we are no longer using a binary code. We're using three symbols: 0, 1, and a space. In an electric signal, how do we distinguish between "off"—0—and a space?
- ▶ The codewords themselves must tell us how to divide up the stream of bits. As we receive the bits, we need to be able to tell when we get to the end of each codeword. For this to occur, our code must satisfy the *prefix-free condition*: No codeword can be the first part (prefix) of any other codeword.

- ▶ Here's a new **prefix-free code**: a is 0, b is 10, c is 11. Consider the same stream of bits as before: 0011100010....
 - The first 0 must be an a , as is the second 0. Then comes 1, which might be the start of either b or c ; the next 1 tells us that it's a c . The bit after that is a 1, but this time, it's followed by a 0, so that's a b .
 - In this way, the series of bits resolves into a series of codewords: 0-0-11-10-0-0-10..., meaning $aacbaab$

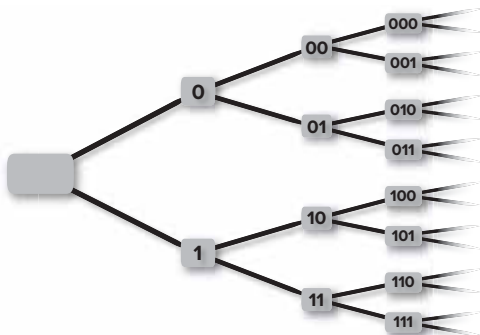
The Binary Tree

- ▶ As we said, no codeword can be the first part of any other codeword. That means that if we have a short codeword, there are many longer codewords that we simply cannot use—all the ones that begin with that short prefix. What does this tell us about the possible lengths of codewords? This question leads us to what mathematicians call a *binary tree*.
- ▶ We start with a point, or node, and from it, we draw two lines off to the right to two new nodes. Then, we do the same for those nodes. We can continue this as far as we like, as shown.
- ▶ We'll say that the first node is at level 0. The next two nodes are at level 1; the four after that are at level 2; and so on. At level L , we have 2^L nodes.
- ▶ Now imagine that these lines are a set of water pipes going from left to right. One unit of water enters the pipe at the first node, and at each node, the water divides into two equal streams. We can label each node by the amount of water (out of that one unit) that reaches it.

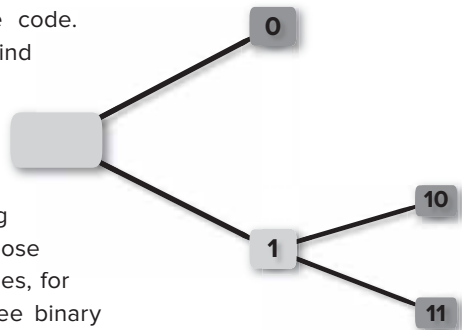




- Each node at level L gets an amount of water $1/(2^L)$, or $2^{(-L)}$. That makes sense, because the one unit of water is spread out equally among all the nodes at that level.
- Here's another way to label the nodes: The first node gets a blank label; the next two nodes are labeled 0 and 1; the next four nodes (level 2) are labeled 00, 01, 10, and 11; and so on. Each node at level L has its own L -bit label.



- ▶ A node's label gives directions for reaching that node. We don't need any directions at the starting point. But at each node, we have to decide whether to continue up or down. Thus, we let 0 stand for up and 1 for down. Node 011 is reached by going up, then down, then down again. And if we go on from that node, all the subsequent labels will start with 011.
- ▶ Now consider the codewords of a prefix-free binary code. Each codeword appears somewhere on the tree. A codeword of length L appears on level L of the tree. But as we travel the tree from left to right, once we encounter a codeword, we can never encounter another codeword further along on the same branch. As far as the code is concerned, once we reach a codeword, that's the end of the tree.
- ▶ For a simple example, consider our three-codeword prefix-free code. We can draw the same kind of diagram for any prefix-free code.



- ▶ Here is the mathematical relation that goes along with the binary tree: Suppose we have a set X of messages, for which we make a prefix-free binary code. The message x has a codeword length, $L(x)$. Then:

$$\sum_x 2^{-L(x)} \leq 1.$$

This is known as the *Kraft inequality*. It is a requirement on the lengths of the various codewords, telling us that we cannot have too many codewords that are too short.

The Kraft Inequality

- ▶ Let's apply the Kraft inequality to the two codes we devised for the three-message source, both the bad code and the good one.
 - In the first code, a is 0, b is 1, and c is 00. The codeword lengths are 1, 1, and 2, and the Kraft sum is $5/4$. But that is not less than or equal to 1. That sum tells us that the code cannot be prefix-free.
 - In the second code, a is 0, b is 10, and c is 11. The lengths are 1, 2, and 2. The Kraft sum here is 1, consistent with the Kraft inequality.
- ▶ If a code is prefix-free, then the Kraft inequality follows. But consistency with the Kraft inequality does not guarantee that the code is prefix-free. A code with codewords 1, 11, and 111 is not prefix-free, but its Kraft sum is $7/8$.
 - However, satisfying the Kraft inequality does guarantee that there is another code, with codewords of the same length, that is prefix-free.
 - If the Kraft sum is less than 1, then we can always shorten some of the codewords to make it equal to 1.
- ▶ Let's suppose we do that, so the Kraft inequality becomes an equation:

$$\sum_x 2^{-L(x)} = 1.$$

This reminds us of a rule of probability: Any probability distribution must add up to 1. We can pretend that those numbers $2^{-L(x)}$ are actually probabilities $q(x)$. They aren't the actual message probabilities, $p(x)$, but some other, fictitious probabilities. We're just making them up out of the codeword lengths.

- ▶ From our made-up probabilities, we can calculate surprises. The length, $L(x)$, of the codeword for x is just the surprise of message x , according to our fictitious distribution $q(x)$.

- ▶ Notice, however: The information inequality tells us that the average q -surprise is at least as great as the actual average surprise, the entropy. That means that the average codeword length can never be shorter than the entropy of the information source.
- ▶ This important principle relates our measure of information—entropy—to the number of bits we need to represent messages. In fact, this principle tells us how we can make the average length close to the entropy.
 - Consider once again our three-message source and the prefix-free code we devised for it. We can summarize what we've found in a table:

x	$p(x)$	$s(x)$	Codeword	$L(x)$
a	0.5	1.0	0	1
b	0.25	2.0	10	2
c	0.25	2.0	11	2

- Notice that for each message, the codeword length equals the surprise. Thus, the average length equals the entropy—in this example, 1.5 bits.
- ▶ This does not always work so perfectly. Let's suppose we have five messages (a through e), with probabilities and surprises shown in the following table:

x	$p(x)$	$s(x)$
a	0.3	1.74
b	0.3	1.74
c	0.15	2.74
d	0.15	2.74
e	0.1	3.32

- The entropy is about 2.2 bits. The codeword lengths must be whole numbers, of course; thus, we simply round the surprise numbers up to determine some lengths. Then, we can use the tree diagram to construct a prefix-free code with those lengths.
- In the end, our codewords are 00, 01, 100, 101, and 1100. The average codeword length for this code is 2.5 bits, which is only a little more than the entropy of 2.20 bits. The best prefix-free code we can find in any given case is called the **Huffman code**.
- ▶ As we saw in Lecture 3, coding blocks of messages together made the codes even a little more efficient. The same is true in this case; we can bring the average codeword length per message even closer to the entropy, $H(X)$.

Shannon's First Fundamental Theorem

- ▶ Shannon's first fundamental theorem can be stated as follows: The entropy, $H(X)$, of a source—the average surprise—equals the minimum average number of bits necessary to code messages from the source.
- ▶ No code can be more efficient than this, but we can find codes that do the job with about $H(X)$ bits on average. Those highly efficient codes make use of regularities in the messages from the source—using shorter codewords to represent more likely messages, just as in Morse code.
- ▶ This is the basis for an important process called *data compression*. As we have already seen, even in a world where bits are cheap, sometimes we want to use as few of them as possible. The number we use depends on the kind of data.
 - Text information, for example, is often represented using the ASCII code, with 1 byte (8 bits) per letter. But from our discussion in the last lecture, we concluded that the entropy of English is much less than that.

- By considering letter frequencies, we found the entropy to be about 4 bits per letter. By analyzing word frequencies, we estimated the entropy to be about 10 bits per word, or 2 bits per letter.
 - Thus, Shannon's first fundamental theorem—the data compression theorem—tells us that we should be able to find a code that “compresses” English text more efficiently than ASCII. Some computer programs, called *data compression programs*, do just that. Roughly speaking, the program analyzes a data file for its regularities and patterns. Based on the analysis, the compression program devises a code to represent the data more efficiently. The program then creates a new file using the new code, including a dictionary for the code so that the original data can be restored.
- The problem of compressing text data is actually not very difficult. Other types of data, such as images, audio, and video, are much harder to compress effectively.
- The problem is so difficult that Shannon's theory by itself cannot solve it. We need another ingredient.
 - Shannon's first fundamental theorem sets the ultimate limits of data compression. But that compression is assumed to be perfectly faithful; we can always exactly reconstruct every bit of the original data from the compressed version.
 - For some purposes, however, such as images, audio, and video, we don't always need that kind of perfect accuracy. The compressed version might need to contain only some of the original information. What information must be included and what can be safely ignored? The answer depends on the way we perceive the world around us.

TERMS

Huffman code: A prefix-free binary code that is the most efficient code for a given set of message probabilities. Devised by MIT graduate student David Huffman in 1952.

Morse Code: The telegraph code devised by Samuel F. B. Morse in the 1840s. (The familiar modern version is a later revision of Morse's original.) Letters are represented by sequences of short and long electrical pulses (dots and dashes), and shorter sequences are used for more common letters. Thus, E is a single dot, while Z is dash-dash-dot-dot.

prefix-free code: A code in which no codeword is the initial segment of any other codeword. Thus, if one codeword is 011, no other codeword can begin with 011... .

READING

Gleick, *The Information*, chapter 5.

QUESTIONS

- 1 If you often send text messages or participate in online chats, you will find yourself using short abbreviations for common phrases. More likely messages use shorter codewords! What are some of the most common abbreviations you use?
- 2 We can turn Morse Code into a true binary code by writing a dot as 10, a dash as 110, and the space between letters as 0. (Roughly, 1 means “on” and 0 means “off” for the electrical signal.) Is this code prefix-free? Calculate the Kraft sum. Find the binary codewords for the most common English letters.
- 3 What happens when you apply a data-compression program, such as gzip, to a file that has already been compressed—that is, already the output of gzip? What if we continue to compress the file over and over again?

Claude Shannon's first fundamental theorem, the principle of data compression, tells us how far we can squeeze information in a binary code: The number of bits per message can be made as low as the entropy of the source—but no lower. Yet sometimes we need to go lower, and for some kinds of data, we can. In this lecture, we'll explore the idea of compression applied to image data, sound data, and video data.

Representing Image Data

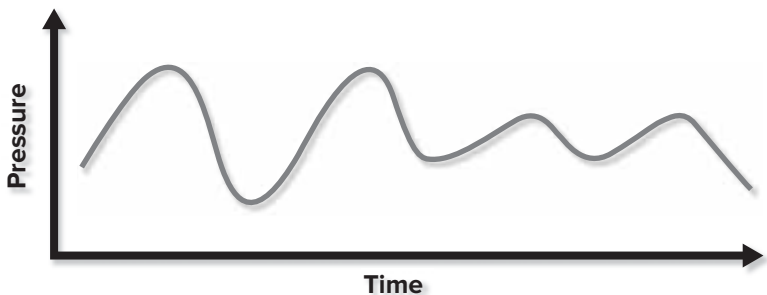
- ▶ **Run-length encoding** is a method for representing image data efficiently. Consider a simple black-and-white cartoon, in which each pixel can be encoded as 1 bit: black is 0 and white is 1.
 - To represent this picture, we would list all the pixel bits, starting in the upper-left corner and going row by row down the picture. Because there are 200×200 , or 40,000 pixels, we can represent the picture by 40,000 bits.
 - However, when we look closely at those 40,000 bits, we find that there are long stretches of pixels that all have the same value. The first several lines are nothing but 0s. Further down, we find dozens of 0s and dozens of 1s together. This pattern is common in simple cartoons: Nearby pixels are often alike.



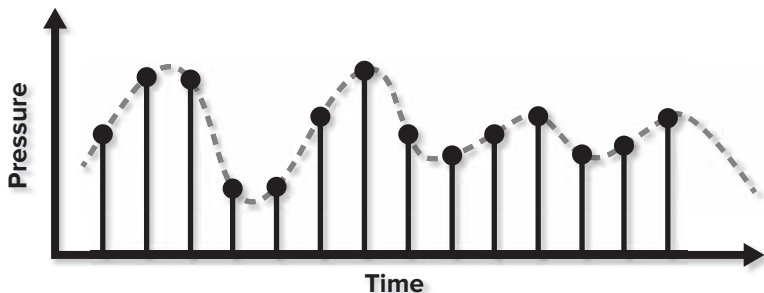
- ▶ In run-length encoding, instead of recording all the 0s on the first line, we record only the number of 0s: 200. We continue in the same way for each line, recording the number of 0s, then the number of 1s, then the number of 0s, and so on, until we get to 200, the end of the line. Some lines need only one number; some lines need several.
- ▶ For this picture, we need just 900 numbers to do the job. Each of the numbers can be written using 8 bits, because 200 is 8 bits long in binary. That means we can represent the image with 8×900 , or 7200 bits—less than one-fifth of the number needed if we coded the image pixel-by-pixel.
- ▶ Run-length encoding is just one of the techniques that have been used to represent image data more efficiently. It compresses simple line drawings and cartoons to a fraction of their original size, but it doesn't work well with photographs or other images with fine textures and subtle variations in color. The methods used to compress those images are quite different, not only in detail but also in their basic philosophy.

Representing Sound

- ▶ Sound is a simpler kind of information than visual data, which is made up of different colors and intensities of light coming from different directions. In contrast, sound waves reach the ear as small, rapid variations in air pressure over time. We could represent a sound perfectly by a graph of pressure versus time:



- ▶ To turn that curve into digital data—data that can be represented by bits—involves two processes: **sampling** and **quantization**.
- ▶ In sampling, we record only the value of the sound signal at certain discrete points in time.
 - Telephone systems generally sample about 8000 times per second, which is enough to capture intelligible human speech, but the resulting quality is not very good. For music, we want a larger sample.
 - Our hearing can only detect sound waves with frequencies less than 20,000 Hertz, or 20 kHz. As we will see in Lecture 9, this means that we only need to record the wave values 40,000 times per second (40 kHz), just twice the frequency. Most audio for music is sampled at 44.1 kHz.
 - We represent the sound wave as a series of discrete wave values—44,100 values per second.

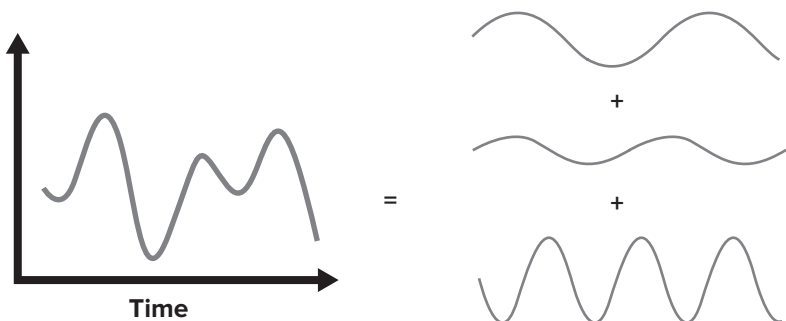


- ▶ Quantization has to do with how accurately we record the sound wave value at each moment. Just as we replaced the continuous time axis with a discrete set of sampled times, now we replace the continuous pressure axis with a discrete set of recordable pressures. For musical sound, we often use 16 bits of information to record the sound signal at each moment. That gives us $2^{16} = 65,536$ different possible levels.

- ▶ The original sound wave becomes 44,100 samples per second, each one consisting of 16 bits of sound-level information. That's more than 700,000 bits per second. In a recording studio, that value is doubled. One minute of recorded stereo music takes up more than 80 million bits, or 10 million bytes. A compact disc, which can store hundreds of millions of bytes, can hold a little more than one hour of music.

Compressing Sound Information

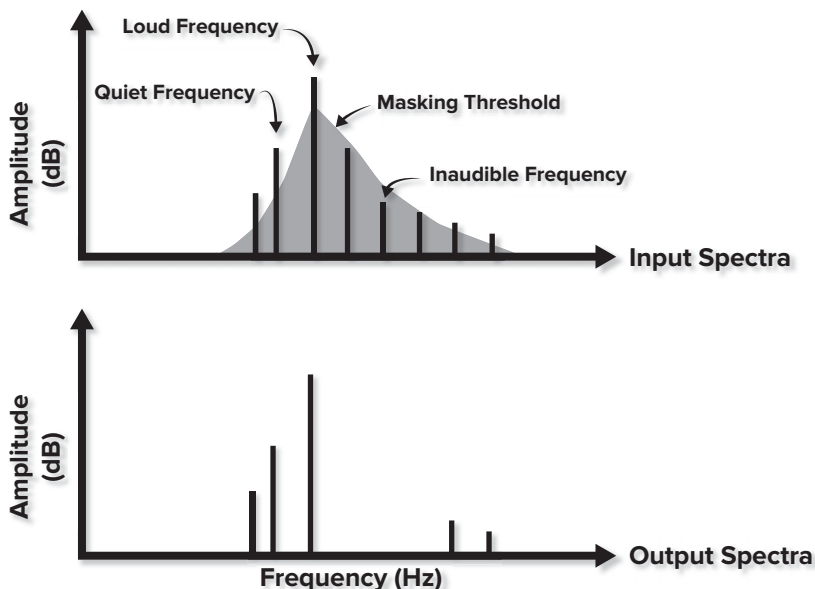
- ▶ We think about sound as a signal—air pressure—that varies over time. We describe it by a series of signal levels at different times. But we can also think about sound as a combination of frequencies.
- ▶ In 1822, the French physicist and mathematician Joseph Fourier showed that any wave form can be formed by combining sine waves—smooth waves of a single well-defined frequency. We can describe the wave by telling how much of each sine wave goes into the mix.



- ▶ Mathematically, we transform a wave from a function, $A(t)$, giving the amplitude of the wave at each time t , to another function, $\hat{A}(f)$, giving the amplitude of the wave at each frequency f . That's called the **Fourier transform**.

- ▶ We can use bits to represent sound as a signal in time or as a group of amplitudes for different frequencies. That by itself doesn't save us anything; because there are as many different frequencies as there are samples in the original sound, we need just as many numbers. But it does allow us to make use of some of the oddities of human perception.
- ▶ When we record and listen to music, we really aren't that interested in reproducing the actual sound—the exact curve of air pressure over time. What we want to reproduce is the human perception of the sound. Only some of the information in the sound makes a difference to our experience of it. Perhaps, then, we can use bits to represent only the information that makes a difference and ignore the rest.
- ▶ This is the idea behind **perceptual coding**, which was proposed in the early 1970s by the German acoustical physicist Manfred Schroeder. The idea is very much in keeping with Shannon's original insight into information: Information is about the distinction between messages. Perceptual coding tries to preserve the distinction between subjective perceptions but not necessarily other distinctions.
- ▶ For sound, perceptual coding is based on the phenomenon of *masking*—the fact that one sound can prevent us from perceiving another, even if the first is still present. Masking can occur between different frequencies and between different times. A loud sound can prevent us from perceiving a softer sound that occurs just after it or, indeed, just before it. Thus, we have both frequency masking and temporal masking.
- ▶ Perceptual coding for sound is based on a mathematical model of human perception, including masking effects. The sound is analyzed using the Fourier transform, and the parts of the sound that will be perceived clearly are identified. Those parts are carefully coded, using a great number of bits. Other parts of the sound are somewhat masked; thus, the coding represents those parts crudely, with only a few bits.
- ▶ One common example of the use of perceptual coding is the **MP3** sound format, which is a complex code for sound that is often used in portable music players. MP3 uses perceptual coding to achieve an amazing

degree of data compression. Recorded music in its uncompressed form uses about 10 million bytes—10 megabytes—per minute. An MP3 version uses about 1 megabyte per minute.



Compressing Visual Data

- ▶ Perceptual coding is also the key to compressing visual data. A good-quality digital photograph contains millions of pixels, and the information contained in 1 pixel is much more than 1 bit. Each pixel can be any color and any brightness.
 - To describe a visual color, we specify three numbers: the intensities for the three basic colors (red, green, and blue). In one common system, each of these three numbers is 8 bits long—1 byte—so that the red, green, and blue levels are between 0 and 255. Red, for instance is (255,0,0)—full intensity for red but not for green or blue. Yellow, a mixture of red and green, is (255,255,0).

- In this way, we can represent color by 3 bytes, or 24 bits of information. That gives us 2^{24} , or about 16 million, possible colors.
- Because human vision can actually distinguish only about 10 million distinct colors and shades, the 24-bit system is almost always sufficient.
- ▶ To achieve compression of visual data, we turn again to perceptual coding. But for visual data, masking is not the key factor. More important is to understand how we perceive detail in an image, and again, the way to approach that question is to use the Fourier transform. For our purposes, we'll consider images that have just one number per pixel—the intensity of light at that point. These are pictures that are made up of lighter and darker shades of gray.
- ▶ An image is essentially a function—light intensity—of two variables, the x and y coordinates of the pixels within the picture. What does it mean to take the Fourier transform of something like that?
- ▶ A sound wave is one-dimensional; it is a function of one variable, time. That wave is made up of simple functions, sine waves, each with a different frequency. The Fourier transform gives the amplitude for each frequency; thus, that is another one-dimensional object—amplitude as a function of frequency.
- ▶ We want to think of our two-dimensional image as made up of simple “sine-wave” pictures, except that we will need two frequencies, one for x and one for y .
 - If both frequencies are 0, the sine-wave picture will just be a uniform, featureless gray color. If we go to a higher x frequency, the image will have a repeating pattern of bands from left to right. If we add a y frequency, the image will have a repeating pattern from top to bottom.
 - It turns out that any picture can be expressed as a combination of sine-wave pictures. Each combination of spatial frequencies will have its own amplitude. And that description of an image in terms of its two spatial frequencies is the Fourier transform of the image.

- ▶ If we examine the Fourier transform of an image, the first thing we find is that for most meaningful images, the amplitudes for the lower spatial frequencies are larger. Those low frequencies represent the overall arrangement of the image. The fine detail of the picture lies in the high frequencies.
 - In low-frequency sine-wave pictures, nearby pixels have nearly the same value.
 - To get differences between nearby pixels—what we need to make fine detail—we must include the sine waves with higher frequencies.
- ▶ If we take the Fourier transform of an image, set all the high frequencies to 0, and rebuild the original, the image becomes blurry. The overall arrangement is the same, but all the fine detail is lost. The fine detail affects our perception of the image, but it does not affect our perception as much as the overall arrangement. Perceptual coding is all about representing only the differences that we perceive.
- ▶ The essential idea here goes back to the concept of quantization, in which we turn a continuous quantity into a finite number of bits. We then decide how many different levels or possible values we will be able to record. The greater the number of different levels, the more bits we will have to use to represent the quantity.
 - Thus, we will use a large number of bits to encode the low-frequency amplitudes in an image. The overall arrangement of the image will be represented exactly.
 - For the higher-frequency amplitudes, we can get by with fewer bits. We can allow ourselves to be more approximate with the details.
- ▶ This is the essential idea behind the widely used **JPEG** format for digital images, and once again, the degree of data compression that can be achieved is astounding. A JPEG version of a photograph may be 10 times smaller than the uncompressed image—1/10 the number of bits—but essentially indistinguishable to the human eye.

Compressing Video Data

- ▶ If data compression is important for images, it is indispensable for video. A single frame of high-definition video contains about 2 million pixels, each with 3 bytes of color information. Thirty video frames are displayed per second, which translates to 180 million bytes of information per second.
- ▶ At that bit rate, even the most advanced optical discs could store only a few minutes of video data. To address this challenge, a series of powerful and flexible image-coding formats (**MPEG-1**, MPEG-2, and so on) was designed.
- ▶ What tricks can we use to compress video data?
 - First of all, because the frames of a video are pictures, we can use the perceptual coding techniques we've already discussed to compress them.
 - In addition, we can take advantage of the repetitive nature of video data. Each video frame is different from the one before, but the differences tend to be small. Another strategy to compress video data, then, is to store only the differences between frames. Every so often, a *key frame* is encoded completely, but for the following frames, only the differences are recorded—using fewer bits.

Encoding and Prediction

- ▶ All this illustrates a fundamental principle about coding and information: The more we can predict, the less we have to encode.
- ▶ In our last lecture, we discussed how to code messages from a source into binary codewords. In an efficient code, the length of the codeword is related to the surprise of the message. A more likely message has a lower surprise and can be represented by fewer bits.

- ▶ If we could somehow improve our ability to predict the next message, it would mean that one or a few messages would have a much higher probability, and all the other possibilities would become much less likely. The entropy of the next message—the average surprise—would become less, and we could use fewer bits on average to record it. This is true whether the messages are the letters of an English text or the frames of a video.
- ▶ Using all these techniques and a few more, we can compress video data by a factor of 20 or more, with almost no noticeable degradation or loss of quality. The key word here is *noticeable*. Perceptual coding is a significant advance in data compression, but it works by changing the problem. Rather than faithfully representing the original message—a sound, an image, or a video—we identify which parts of the original information actually affect our subjective experience. We then put our efforts into conveying those parts and discard the rest.
- ▶ We have seen two different concepts of data compression.
 - One concept is like Huffman coding for the letters of the alphabet or run-length encoding of simple black-and-white cartoons. From the compressed version, we can exactly recover the original; nothing is lost. This is *lossless data compression*.
 - The other kind of compression is used in compressing sound with MP3 encoding or an image with JPEG. Here, we represent only part of the information, and the exact original can no longer be recovered. This is *lossy data compression*. We deliberately discard information as the price we are willing to pay for increases in efficiency.

TERMS

Fourier transform: A way to mathematically represent a signal that varies in time or space as a combination of different frequencies, discovered by Joseph Fourier in 1822. A basic tool in perceptual coding of both audio and image data and important for understanding the bandwidth of an analog signal.

JPEG: A flexible and highly efficient data-compression format for many kinds of image data, introduced by the Joint Photographic Experts Group in the 1990s. There is relatively little noticeable loss of image quality except for cartoons and diagrams involving fine lines and lettering, which tend to be surrounded by unwanted “artifacts” in the coded picture.

MP3: An audio data-compression format, originally developed as part of a system for data compression of video. (*MP* stands for “moving picture.”) MP3 takes advantage of the masking phenomenon of human audio perception to greatly reduce the number of bits needed to transmit music or speech. For instance, 10 megabytes of uncompressed musical sound on an audio CD might occupy only 1 megabyte in MP3 format.

MPEG: A group of data-compression formats for video data, created by the Moving Picture Experts Group beginning in the late 1980s. In addition to techniques of image compression, the MPEG formats take advantage of the fact that successive frames of a video usually have only small differences. One form of MPEG-4, also known as the H.264 format, is the usual standard for streaming video and Blu-Ray disks.

perceptual coding: A powerful strategy for compression of media data, such as audio, still images, and video, by making a faithful representation of the subjective experience of the data, while losing information in ways that can be entirely unnoticeable. Pioneered by Manfred Schroeder in the 1970s.

quantization: The conversion of an analog quantity (such as a voltage) to a digital quantity, one with only a finite number of representable values. For example, the sample values for sound in an audio CD are represented by 16 bits, giving 65,536 possible sound levels at each moment.

run-length encoding: An efficient technique for encoding simple images, in which the number of contiguous identical pixels is given. Not very efficient for photographs or other images with complex textures and fine variations in shading.

sampling: The representation of a continuous signal (such as a sound wave) by a series of discrete values. According to the Nyquist-Shannon sampling theorem, a sampling rate of twice the signal bandwidth is both necessary and sufficient to represent the signal faithfully. Because human hearing reaches to a frequency of only around 20,000 Hz, audio CDs sample the sound at 44,100 samples per second. This is enough to reproduce the audible sound.

READINGS

Cook, ed., *Music, Cognition and Computerized Sound*.

The Internet is the place for the best sources about MP3, JPEG, MPEG, and other media data formats. Start with the online encyclopedia Wikipedia (which is usually quite good on computer science subjects). Programmers may wish to follow links to official standards documents that give details of the various compression algorithms.

QUESTIONS

- 1 Detectives in movies are sometimes presented with a blurry, low-resolution computer image of an unknown subject. With a few keystrokes, they inevitably “enhance” the image and arrive at a much sharper picture. Comment on the basic realism of this trope.
- 2 We mentioned a piano as a kind of “natural” Fourier transform. Can you think of any other musical instruments that might do the same thing?

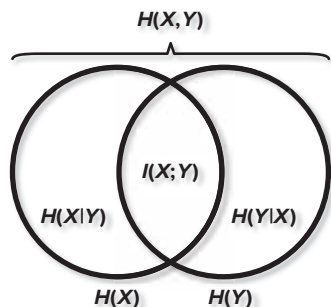
Noise is ubiquitous, and it produces ambiguity. With noise, the message we receive may not tell us everything about the message that was transmitted. Some information may be lost, introducing the possibility of error. How do we communicate in the presence of noise and still avoid errors? This is one of the most important questions in the science of information, and it's the question that Claude Shannon explored in his path-breaking paper in 1948, in which he established the foundations of information theory. In this lecture, we'll look at Shannon's remarkable and surprising answers.

Communication Channels

- ▶ A *communication channel* is any means by which information can be conveyed—anything with an input and an output. For instance, a fiber optic line, a radio link, or sound waves through the air can all be channels. The concept of a channel is not only about carrying information from place to place; it also can represent the storage and retrieval of information over time.
- ▶ If the output were always exactly the same as the input, then the idea of a channel would be trivial, but that is not usually the situation. The channel establishes the relationship between input and output. For each possible input, the channel determines which outputs are possible and with what likelihood they may occur. Thus, the way to analyze a channel is to use probability.
- ▶ The input is an information source, X , whose messages, x , have probability $p(x)$. The set of possible outputs is Y . For some channels, the output is exactly determined by the input. But other channels act in a less predictable way, and several possible outputs might result from a given input.

- ▶ To describe the channel, we need to describe the rule by which the inputs lead to the outputs. That rule is specified by conditional probabilities $p(y|x)$ —the probability that the output, y , is produced given the input, x .
- ▶ The important lesson here is that the communication process involves both an input and an output, which are related to each other but are not necessarily the same. We know the probabilities, $p(x)$, of the various possible inputs, and the channel determines the conditional probabilities, $p(y|x)$.
 - The **conditional probability** $p(y|x)$ is just the joint probability $\frac{p(x,y)}{p(x)}$.
 - We can turn that around and say that the joint probability $p(x,y)$ is $p(x) \times p(y|x)$.
 - The probability of an input-output pair is simply the probability of the input multiplied by the probability of the output given the input.
- ▶ The Shannon measure of information is the entropy, the average surprise.
 - The input has an entropy, $H(X)$, which is the total amount of information in the transmitted message. The output also has an entropy, $H(Y)$.
 - The combination of the input and output has a joint entropy, $H(X,Y)$, which we can calculate from that joint distribution $p(x,y)$.
- ▶ For any two variables with any joint probability distribution, the joint entropy is no larger than the sum of the individual entropies. This makes intuitive sense. The information in both variables cannot be any larger than the information in one added to that in the other. And it might be less.
 - If the input is a 1-bit message and the channel is perfect, the output is always exactly the same as the input: $H(X) = 1$ bit, and $H(Y) = 1$ bit.
 - But since knowing X automatically tells us Y , $H(X,Y)$ is also 1 bit. That's much less than the sum of 2 bits.
- ▶ Can we ever have $H(X,Y) = H(X) + H(Y)$? Yes, we can, provided the two variables are totally independent of each other. In that case, $p(x,y) = p(x)p(y)$ for every pair of (x,y) values.

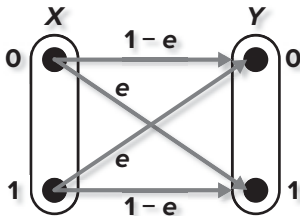
- ▶ Before any communication occurs in the channel, the receiver knows neither the input, X , nor the output, Y —neither what is sent nor what is received. Thus, the amount of information the receiver lacks at the outset is $H(X,Y)$.
 - During the communication process, the receiver learns the value of the output, Y , and gains an amount of information, $H(Y)$.
 - But that still leaves something left over—the information that the receiver lacks about X even though he knows all about Y . Let's call that the **conditional entropy**, $H(X|Y)$.
 - That's how much information the receiver does not learn about the transmitted message. It's a measure of failure to communicate. Information theorists sometimes call the conditional entropy the *equivocation* of the message. It tells us how much the receiver isn't sure about X , even when he or she knows Y .
- ▶ A more positive measure would be the amount of information that the receiver gains, which is the difference between $H(X)$ and $H(X|Y)$. This important quantity is called the **mutual information**.
- ▶ The diagram here sums up these concepts.
 - The left circle represents $H(X)$, the amount of information in the input. The right circle is $H(Y)$, the amount of information in the output. The whole area is $H(X,Y)$, the information in both.
 - $H(X|Y)$ is the part of X that is not included in Y —the information still missing about X even when Y is known. We can see that this area is $[H(X,Y)] - H(Y)$. $H(X|Y)$ is the same on the other side.



- The overlap of the two circles is the mutual information $I(X;Y)$. That's the Y information that is also the X information—the amount that the output of the channels tells us about the input. That's how much information the channel actually conveys to the receiver.

Binary Symmetric Channel

- Let's illustrate these rather abstract concepts with a simple example. Suppose that the input X is a bit—possible messages 0 and 1, each with equal likelihood $1/2$. The output is also a bit. However, the output might not agree with the input; the bit might “flip”—change to the opposite value—along the way. We let the number e represent the probability that an error of this kind occurs. This situation is shown in the diagram below:



$$p(0|0) = 1 - e$$

$$p(1|0) = e$$

$$p(0|1) = e$$

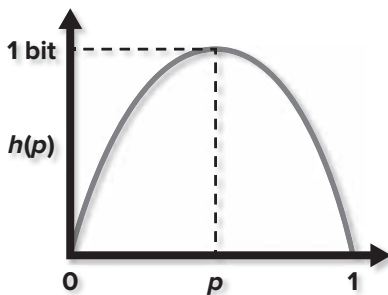
$$p(1|1) = 1 - e$$

- The probability that input 0 will yield output 0 is $1 - e$, while the probability that input 0 will yield output 1 is e ; and so on.
- This is called the **binary symmetric channel**—*binary* because its inputs and outputs are just binary digits and *symmetric* because the probability of error is the same whether the input is 0 or 1.
- This represents what happens with two people, Alice and Bob, talking at a noisy party. X is what Alice says—yes or no, 1 or 0. Y is what Bob thinks he hears. That might be the same as what Alice says, but there is some probability, e , that he hears wrong. The error probability, e , is determined by the level of noise at the party.

- What about the entropies? $H(X)$, the entropy of the input, is 1 bit. $H(Y)$, the entropy of the output, is also 1 bit. $H(X,Y)$, the joint entropy of input and output, is a little trickier to calculate.
 - We first write down a table of the joint probabilities:

		Y	
		p	
X	0	$\frac{1}{2}(1-e)$	$\frac{1}{2}e$
	1	$\frac{1}{2}e$	$\frac{1}{2}(1-e)$

- Applying Shannon's formula to these probabilities involves a little algebra. Skipping to the answer, we get the result for the joint entropy: $1 + h(e)$ bits, where h stands for the binary entropy function of the probability e .
- The binary entropy function is easy to explain: The quantity $h(p)$ is just the entropy of a binary source with message probabilities p and $1-p$. Below is a graph of this function as p varies from 0 to 1.



- ▶ The binary entropy is greatest when $p = 1/2$ —that is, when the two messages are equally likely. In that case, the entropy is 1 bit. The entropy goes to zero if p is either 1 or 0.
 - In the one case, message 0 is certain, and in the other case, message 1 is certain, but in either case, there can be no surprise; thus, the average surprise is zero.
 - The whole curve is symmetric; as far as the entropy is concerned, it does not matter whether 0 or 1 is a more likely message.
 - Finally, note that the curve of the binary entropy function is very steep at either end. Even if $p = 0.9$ —90% of the way over to 1—the value of $h(p)$ is still surprisingly high: 0.469 bits—almost $1/2$.
- ▶ Now that we know that the joint entropy is $1 + h(p)$ bits, we can find out other parts of our Venn diagram.
 - For instance, we know that $H(X|Y)$ —the amount of information Bob still lacks about X after he receives Y —is just $h(e)$ bits.
 - The mutual information—the amount of information that is actually conveyed by the channel—is $1 - h(e)$ bits.

Extreme Examples

- ▶ To help us understand all this, let's think about a couple of extreme situations. First, suppose that there is no noise at all and $e = 0$. Then, $h(e) = 0$. Whatever Alice says, Bob hears. The mutual information—the amount that Alice actually conveys to Bob—is 1 bit.
- ▶ Now suppose that the party is so noisy that Bob cannot tell at all what Alice is saying. He is just as likely to be wrong as right about her answer—an error probability of $e = 1/2$. Then, $h(e)$ has its maximum value, 1 bit. That makes the mutual information zero. No information is conveyed.
 - When the output of the channel is completely independent of the input, the mutual information is zero, and the channel conveys no information.

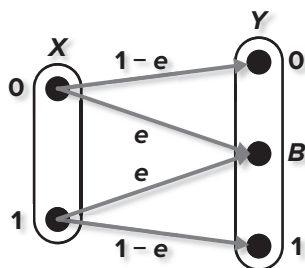
- That's what happens when Bob's error rate is $1/2$. With that much noise, Bob's guesses about Alice's message will be no better after the communication than before. He has gained no information.
- ▶ Now consider a situation in which Alice and Bob are communicating through some machine that distorts the words that are said. Whenever Alice says, "no," it comes out sounding like "yes" and vice versa. This is the case where $e = 1$, and at first, this seems even worse than before. But the binary entropy function is 0; thus, the mutual information is 1 bit. How can that be?
 - We always assume that Alice and Bob know how their communication channel works—how it distorts the inputs. To put it another way, the probabilities that describe the channel are their probabilities—their understanding of how it works. Thus, they know the value of e . In the distortion-machine situation, Bob knows that whenever he hears "yes," Alice must have said "no" and vice versa. In other words, this channel is just as good as a perfect one.
 - Information is, after all, about the distinction between messages, not how those messages are represented. The distorting channel still preserves the distinction between messages, because different inputs still lead to different outputs. No information has been lost. The code has simply changed.
- ▶ For all the in-between situations, where e is not 0, $1/2$, or 1, we simply need to calculate $h(e)$ and apply the formulas we've found. Thus, if e is $1/10$, then $h(e) = 0.469$ bits. The mutual information—the amount of information that gets through—is, therefore, 0.531 bits.

Communication Errors

- ▶ Is it true that a 10% chance of error destroys not 10% of the information but almost half of it? This seems counterintuitive at first, but it is actually quite reasonable, once we consider what an error really entails.

- ▶ In a classroom situation, information goes into students over the course of the semester, and later, on an exam, information comes out. But students sometimes make errors; thus, there is some noise in the communication channel. Mathematically, for each question on any exam, we can say that the input, X , is the original fact, and Y is the answer on the exam.
- ▶ One student, Charlie, typically gets about 90% of the questions on a true-false exam correct. If he misses 10 questions on a 100-question exam, his error probability is $e = 0.1$. Is Charlie missing 10 bits of information or more than 10?
 - For Charlie to correct all his mistakes, the teacher would have to tell him where all his errors occurred. We can estimate how many bits of information this is.
 - The grading marks Charlie receives (x's for wrong answers and checkmarks for correct answers) are like the output of a binary information source—the “error source”—with a probability of $1/10$ (the value e) for an x and $9/10$ ($1 - e$) for a checkmark.
 - The binary information function, $h(e) = 0.469$, bits tells us the entropy per grading mark.
 - Shannon's first fundamental theorem, the data compression theorem, tells us that the teacher could represent that grading information using about $h(e)$ bits per message in the long run. Thus, the teacher could correct Charlie's mistakes by providing him with about $h(e)$ additional bits of information per question—around 47 bits for the whole exam.
- ▶ Compare Charlie to another student, Diane. She also gets about 90% of the questions right. However, she works a little differently from Charlie. When she thinks she knows the answer, she invariably answers correctly. But when she doesn't know the answer, she just leaves the question blank. On the exam, she leaves about 10 of the 100 questions blank.

- Diane is a very different channel. She has three outputs: true, false, and blank. We could call these 1, 0, and B. Diane's errors are always ones in which she responds with a B instead of 0 or 1. This is called a **binary erasure channel**, which has a slightly different diagram:



$$p(0|0) = 1 - e$$

$$p(B|0) = e$$

$$p(1|0) = 0$$

$$p(0|1) = 0$$

$$p(B|1) = e$$

$$p(1|1) = 1 - e$$

- Because the two inputs are equally likely, the table of the joint distribution is easy to figure out.

		p		
		0	B	1
X	0	$\frac{1}{2}(1-e)$	$\frac{1}{2}e$	0
	1	0	$\frac{1}{2}e$	$\frac{1}{2}(1-e)$

- The input entropy, $H(X)$, and the joint entropy, $H(X,Y)$, are the same as before, but the entropy for the three-way output, $H(Y)$, is not the same. The equivocation, $H(X|Y)$, turns out to be just e bits, and the mutual information is $1 - e$ bits per question.
- Thus, Diane really is missing just 1/10 of the information on the test. She knows when she doesn't know the answer to a question, which means she actually knows a good deal more than Charlie, who thinks he knows the answer but is sometimes wrong. To correct Diane, the teacher only needs to give her 10 additional bits of information.

Summing Up Communication Channels

- ▶ A communication channel takes an input X and yields an output Y . The two might not always be the same. We describe the channel by some conditional probabilities, the $p(\text{output}|\text{input})$. If we put that together with the input probabilities, we find the joint distribution over inputs and outputs. That lets us calculate entropies $H(X)$, $H(Y)$, and $H(X,Y)$.
- ▶ From these, we have defined two important new quantities:
 - The conditional entropy $H(X|Y) = H(X,Y) - H(Y)$, also called the equivocation, which measures how much of the input information is not conveyed by the output Y .
 - The mutual information $I(X;Y) = H(X) - H(X|Y)$, which measures how much input information is conveyed by the output Y .
 - These correspond to different areas in our Venn diagram.
- ▶ For the binary symmetric channel, $H(X|Y) = h(e)$, the binary entropy function of the error probability e . And that means that the mutual information $I(X;Y) = 1 - h(e)$.
- ▶ What sort of errors we have matters very much. In the binary symmetric channel, like Charlie's true-false exam, the errors look no different from correct bits. That means it takes more additional information to correct the errors. In the binary erasure channel, like Diane's exam, the errors are self-announcing, and it takes less additional information to correct them.

Correcting Errors

- ▶ In most situations, we don't have a second channel that is perfect, such as a teacher, to correct errors. For example, when a spacecraft sends a weak radio signal from billions of miles away, there is no noiseless second channel to help us fix the parts of the signal that have been lost or distorted. What can we do about errors in such situations?

- ▶ One way we can reduce the overall probability of error is to send messages redundantly. In effect, we are sending a longer message, and different parts of that message help us correct errors in other parts. No part is immune from error, and there is always a chance that we could make a mistake, but we can at least improve the likelihood that we will get the message right.
- ▶ This improvement, however, comes at a cost. We use the channel a number of times to send a fairly simple message, which uses many resources and slows down communication. And to be even more certain that we will avoid error, it seems that we must be even more redundant, repeating the message more times. Does this mean that the only way to avoid error would be to communicate extremely slowly? Is there an inescapable trade-off between the volume and the reliability of information?
- ▶ Shannon regarded this as the key issue in information theory. In thinking about the problem, he asked: What is the greatest measure of information that a given communication channel can convey? That is called the capacity, C , of the channel, defined as the maximum of the mutual information $I(X;Y)$ over all choices of input message probabilities: $C = \max I(X;Y)$.
- ▶ We've calculated the mutual information for the binary symmetric channel and the binary erasure channel, and it turns out that those were maximum values for each channel. Thus, we have already calculated their channel capacities.
- ▶ Here is what Shannon proved: As long as you don't try to exceed the capacity of your channel, you do not have to communicate more slowly to get rid of errors.
 - Suppose you decide on an information rate, R —a number of bits to send per use of the channel—so that $R < C$, the **channel capacity**.
 - Then, by coding many messages together (block coding) and by designing the right code and decoding procedure, we can achieve two results simultaneously: (1) sending an amount of information, R , each time we use the channel, and (2) making the overall probability of error go to zero as we use larger and larger blocks in our code.

- ▶ Shannon also showed that if we try to send too many bits each time—more than C —then in the long run, the error probability goes to 1. We're sure to make mistakes. This is called *Shannon's second fundamental theorem*.
- ▶ This theorem tells us that we can defeat noise without paying too high a price. There is no inescapable trade-off between information volume and information reliability. The possibility of error does not force us to communicate extremely slowly. We can do error correction efficiently, and the channel capacity tells us just how efficiently.
- ▶ Shannon's second fundamental theorem tells us that effective and efficient error-correction techniques exist, but it does not tell us what they are. However, in the years since the theory, we have learned a great deal about practical error correction. We'll turn to this topic in the next lecture.

TERMS

binary erasure channel: A simple communication channel with two possible inputs (0 and 1) and three possible outputs (0, 1, and B, for “blank”). The only possible errors replace the input bit with B. This is analogous to a student filling out a true-false test but sometimes leaving an answer blank.

binary symmetric channel: A simple communication channel with two possible inputs and two possible outputs, both of which may be designated 0 and 1. The channel is “symmetric” because the likelihood of an error (changing 0 to 1 or 1 to 0) is the same whether the input is 0 or 1.

channel capacity: The maximum mutual information for a communication channel. According to Claude Shannon's second fundamental theorem, the channel can be used to send any number of bits less than the capacity, with a negligible overall probability of error. (This theorem thus guarantees the existence of effective and efficient error-correcting codes, though it does not show how to construct them.)

conditional entropy: The average entropy based on conditional probability. Thus, in a noisy channel, the conditional entropy of the input message given the channel output is a measure of the information that the receiver still lacks about the message even when the output is known. Conditional entropy is, therefore, a measure of the noise in a channel.

conditional probability: The likelihood of an event given that some other event also takes place.

mutual information: The amount of information that is conveyed by a noisy channel. Mathematically, the difference between the entropy of the input and the conditional entropy of the input given the output. According to Claude Shannon's second fundamental theorem, the maximum of the mutual information for a channel is the channel capacity.

READING

Pierce, *An Introduction to Information Theory*, chapter 8.

QUESTIONS

- 1 Alice sends X to Bob, who receives Y . Show that the amount of information that Bob has about what Alice sent is equal to the amount of information Alice has about what Bob receives.
- 2 Suppose our binary symmetric channel has a very low noise rate: $e = 0.001$. The expected number of errors in a 1000-bit message is about 1. How many bits of information are lost in that message?

Error-Correcting Codes

Optical disks for music and movies, reliable computer operating systems, communication with spacecraft billions of miles away—none of these things would be possible without the process of error correction. It's one of the most amazing—and most underappreciated—stories in the science of information. In this lecture, we'll examine a number of error-correcting codes.

Human Language and Error Correction

- ▶ Human language appears to be very inefficient and redundant. Every human language uses many more symbols than the messages seem to require—more letters of text or more basic sounds. But this **redundancy** serves a vital purpose: It makes our communication much less subject to errors.
- ▶ In English, with 26 letters, the number of five-letter groups that could form words is 26^5 , which is almost 12 million. However, English contains only about 9000 five-letter words.
- ▶ In the vast space of all possible words, some of these five-letter combinations are close together; they might differ by only one letter (*pride* and *prime*). We'll call these words that are just one step apart *neighbors*.
 - The total number of words with a one-letter difference from a starting word—the neighbors of that word—is 125. Only a few of those are actual English words. For instance, the word *grove*, with six neighbors, is about average: *drove*, *glove*, *grave*, *grope*, *prove*, and *trove*.
 - The fact that most words have only a few neighbors makes English resistant to errors. Suppose someone sends you a message consisting of a five-letter word, but an error occurs in one letter.

Because there are 125 possible changes but only 6 (on average) are real words, the probability is better than 95% that the error will be obvious. If the word is part of a larger message, the likelihood is also very high that the context will enable you to figure out the right word in spite of the error.

- ▶ What is true for written text is also true for spoken language. The potential for error in speaking and hearing is quite high, but because English is so redundant, a few misheard phonemes don't usually present a problem. The hearer corrects them unconsciously.
- ▶ Error correction wouldn't work as well if English words were more crowded together in the space of all possible words. Mistakes in one letter or sound would be more likely to turn the correct word into some other plausible word.
- ▶ Given this capability for error correction, it's wrong to conclude that human language is an inefficient code for communication.

The Basis of Error-Correcting Codes

- ▶ In the last lecture, we saw that Shannon's second fundamental theorem guaranteed that error correction is possible, provided that the communication channel allowed some information to get through.
 - The amount of information conveyed by a channel is measured by its mutual information, and the maximum mutual information for the channel is its capacity, C . If we are clever about our code and our decoding procedure, then in the long run, we can send up to C bits per use of the channel, while making the overall probability of error as small as we wish.
 - Although Shannon's theorem guarantees that error correction is possible, it does not tell us how to go about it. This problem remains a subject of research in mathematics, computer science, and engineering to this day.

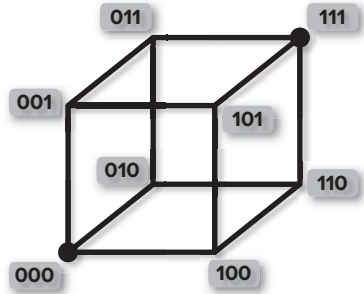
- ▶ Error-correcting codes are based on the same idea we encountered when discussing the redundancy of English. We will use longer codewords, and the codewords in our code will be scattered very sparsely in the set of all possible codewords. In some sense, the codewords will be far apart from each other, which will make it easier to recognize and fix potential errors.

A Simple Error-Correcting Code

- ▶ Let's start with the simplest possible error-correcting code.
 - Alice wants to send a 1-bit message to Bob, 0 or 1, but there is some possibility that an error will be made when the bit is transmitted. For this reason, Alice uses codewords containing 3 bits; she simply transmits the same bit three times: 000 or 111.
 - Because Bob might receive some bits incorrectly, he checks to see if there are more 0s than 1s and decodes accordingly.
 - Sending more bits makes things more redundant, but it also increases the opportunity for errors. Will this method actually improve Bob's chances of decoding Alice's message correctly?
- ▶ Suppose that each bit transmitted has an error probability, e , which we will take to be 5%. That's the probability of error if Alice simply sends her 1-bit message once without using any error correction. To put it another way, the probability that a bit is transmitted correctly is $1 - e$, or 95%.
- ▶ Now consider Alice's "repeat three times" code.
 - The likelihood that Bob receives all three bits correctly is $(1 - e) \times (1 - e) \times (1 - e) = (0.95)^3$, or 0.857.
 - But Bob will also decode the message correctly if just one of the bits is wrong. There are three ways that can happen. The first, second, or third bits could be the wrong one, and each one has a probability $(1 - e) \times (1 - e) \times e$ —two rights and one wrong—which

gives $(0.95)^2 \times (0.05) = 0.045$. The total probability that Bob recovers the original 1-bit message is, therefore, 0.993.

- We can visualize how the code works. The eight possible 3-bit words can be arranged at the corners of a three-dimensional cube. The three binary numbers are coordinates of those points.

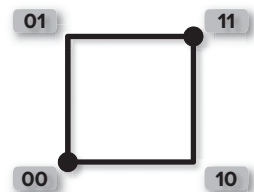


- We can define how far apart two codewords are using the **Hamming distance**, named after Richard Hamming, a pioneer of error-correcting codes.
 - The Hamming distance between two codewords is the number of places in which they differ. Here, 000 has a distance of 1 from 100, a distance of 2 from 101, and a distance of 3 from 111.
 - In the diagram, the distance between codewords can be found by counting the number of edges on the cube we have to travel to get from one to the other.
- The two codewords Alice uses are 000 and 111, which are separated by a Hamming distance of 3. This number is significant.
 - If Alice sends the codeword 000 and an error occurs in 1 bit, Bob receives something like 001. But with one error, the received 3-bit word is still closer to the original 000 (Hamming distance 1) than to the alternative 111 (Hamming distance 2).
 - Because the codewords are separated by Hamming distance 3, any single error will still leave us closer to the original than to anything else.
 - Thus, whatever Bob receives, he shifts it back to the nearest codeword in Alice's code before decoding the message. The whole process goes as follows: encode, then send through the channel, then do error correction, then decode.

- ▶ There are four important facts that we need to know about an error-correcting code to understand how it works:
 - The first is what kind of symbols are being used, such as bits, decimal digits, letters of the alphabet, and so on.
 - The next fact is n , the number of symbols used in the codewords. For Alice's "repeat three times" binary code, $n = 3$.
 - The third fact is k , the equivalent number of symbols being represented by the codeword. In Alice's code, the codewords represent a single bit, either 0 or 1; thus, $k = 1$. Alice is using three symbols to code one symbol.
 - Finally, we need to know d , the Hamming distance between the closest pair of codewords. In Alice's code, there are only two codewords and they are separated by three; thus, $d = 3$.

Error-Correcting Power

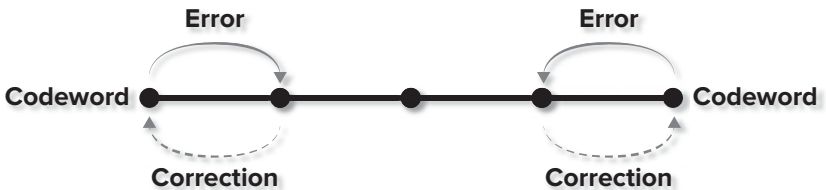
- ▶ Coding theorists often designate a code by its n and k values. They would say that Alice is using a (3,1) code, with minimum distance $d = 3$. That minimum distance tells us what kind of error-correcting properties the code has. For instance, if $d = 1$, there are two codewords that differ in only symbol. That means that a single error could turn one codeword into another; for instance, *grove* could become *grave*.
- ▶ Suppose $d = 2$. We can illustrate this with a 2-bit code on a square, as shown here. It's a (2,1) code, and the two codewords are 00 and 11. If Alice sends 00 and one bit picks up an error, Bob might receive 01. He cannot correct that error, because 01 is equally close to 00 and 11. If Bob knows that an error has occurred, however, that may be enough. He can notify Alice that her message did not get through.



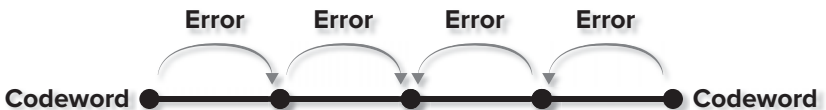
- ▶ We've already seen that having $d = 3$ enables us to correct any single error, because a single error still leaves us closer to one codeword than any other.
- ▶ What about $d = 4$?
 - Imagine two codewords separated by a distance of 4.

Codeword ● ——— ● ——— ● ——— ● ——— ● **Codeword**

- We can correct any single error in the codewords, because that error would only shift the received message by a Hamming distance of 1. There is no ambiguity about which original codeword is closest.



- If there are two errors, we might wind up at a codeword that is in the middle, a distance 2 from two different codewords. We would not know how to correct such an error, because neither codeword is closest. But we can detect that the fact that two errors have occurred, and that's worth something.



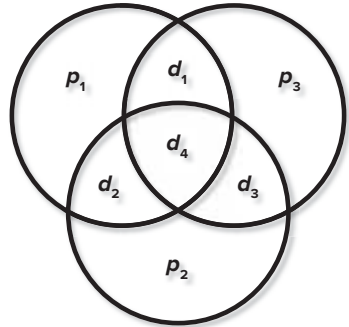
- If the codewords were a little farther apart ($d = 5$), we would be able to correct even two errors, because we'd still wind up closer to the original codeword than to any other.
- ▶ We can summarize the error-correcting power of a code just by the number d , the minimum Hamming distance between codewords:
 - $d = 1$ means no error correction or detection
 - $d = 2$ means no error correction, detect one error
 - $d = 3$ means correct one error
 - $d = 4$ means correct one error, detect two errors
 - $d = 5$ means correct two errors.
- ▶ If d is an even number, we can detect $d/2$ errors and correct one fewer. That means a $d = 8$ code can detect four errors or correct up to three of them. If d is an odd number, we can correct up to $(d - 1)/2$ errors; thus, if the code has $d = 13$, we can correct up to six errors.

The Hamming (7,4) Binary Code

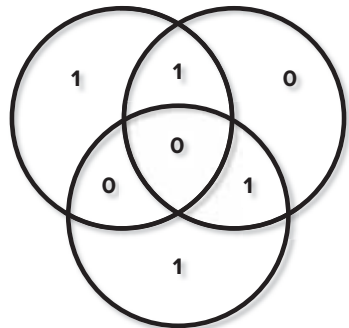
- ▶ In a (7,4) code, $n = 7$, which means that the codewords are 7 bits long; $k = 4$, which means that the codewords represent 4 bits of information. That means we need $2^4 = 16$ different codewords. As we will see, for this code, $d = 3$; thus, the code can correct any 1-bit error.
- ▶ We cannot draw a diagram of a seven-dimensional cube, but it is not too hard to understand how the code works.
 - Each codeword has 4 data bits: d_1 , d_2 , d_3 , and d_4 . These carry the message. In addition, we add three more parity bits: p_1 , p_2 , and p_3 . These are calculated from the data bits using the XOR function, as you recall from the Boolean logic we discussed in Lecture 2.

- As a reminder, $0 \text{ XOR } 0$ and $1 \text{ XOR } 1$ are both 0, while $0 \text{ XOR } 1$ and $1 \text{ XOR } 0$ are both 1. The parity bits are calculated in this way: $p_1 = d_1 \text{ XOR } d_2 \text{ XOR } d_4$, $p_2 = d_2 \text{ XOR } d_3 \text{ XOR } d_4$, $p_3 = d_3 \text{ XOR } d_1 \text{ XOR } d_4$.

- There's a pattern here, but it's a little difficult to understand. We can draw a picture, starting with three overlapping circles, and label each of the seven regions with one of the bits:



- The data bits come from the message.
- Here's the rule for the parity bits: Each parity bit is the XOR of the 3 bits it shares a circle with. You can think of this rule in an even simpler way: The parity bits are set so that each circle contains an even number of 1s. Thus, if the data bits are 1010, the parity bits are 110.



- Because each data bit affects its own value and the value of two parity bits, the distance between nearest codewords is 3. Therefore, the Hamming (7,4) code ought to be able to correct any 1-bit error, in either data bits or parity bits.

- Suppose one of the data bits is flipped, say d_1 . That means that two of the circles in the diagram now contain an odd number of 1s, which tells us that the problem must be in the overlap of those two circles, namely d_1 .
 - If d_4 is flipped, all three circles now have an odd number of 1s, and we know that the three-way overlap is to blame.

- If one of the parity bits is flipped, that only affects the circle containing that bit. In any case, we know exactly where the error occurred, which means we can correct it.
- ▶ If errors are not too frequent—for instance, if double errors and triple errors are extremely rare—then the Hamming (7,4) code will do a good job of correcting them.
 - Suppose the chance of an error in 1 bit is 1 in 1000. In other words, our channel is a binary symmetric channel with error probability $e = 1/1000$.
 - If we are sending a 4-bit message, the chance of an error is about four times as large— $1/250$.
 - But if we use Hamming's code to send those 4 bits using 7 bits, the overall chance of making an error is a lot less—about 2 in 1,000,000.
- ▶ Modern error-correcting codes are much more sophisticated than the Hamming (7,4) code, but they work in similar ways and are used everywhere.

Reed-Solomon Codes

- ▶ Higher error rates need more advanced codes. Among the most popular and powerful types are **Reed-Solomon codes**, developed more than 50 years ago by Irving Reed and Gustave Solomon.
- ▶ In a common Reed-Solomon code, the basic symbols are not bits but bytes—groups of 8 bits considered as a single symbol. The codewords are 255 bytes long, and they represent 223 bytes of data. This is a (255,223) code. For Reed-Solomon codes, d , the minimum distance between codewords, is always $n - k + 1$, or 33 in this case. The (255,223) Reed-Solomon code can correct up to 16 single-byte errors.

- ▶ One of the things that makes the Reed-Solomon codes so useful is their ability to handle bursts of errors. The Reed-Solomon (255,223) code can correct 16 byte errors—and a byte error could be a mistake in 1 or all of the 8 bits in that group. That means that the code can correct a burst of more than 100 consecutive 1-bit errors in the codeword. Through a process of **interleaving** codewords so that the bits of each codeword are spread far apart, the Reed-Solomon (255,223) code can do even better than that.
- ▶ Variations of Reed-Solomon codes are used for error correction on audio CDs and in transmitting images from the Voyager spacecraft back to Earth. Indeed, the improvement in error correction on the Voyager spacecraft was an amazing accomplishment. Scientists and engineers on Earth created codes that were more efficient and more robust against errors to improve the way information from the spacecraft was represented.
- ▶ However, that triumph of coding and error correction is only part of the story. The radio transmitters aboard the Voyager spacecraft are very weak—about 20 watts, yet we are still in communication with them, 12 billion miles from the Sun. How can we get any information at all from such a weak signal? To answer that, we need to explore more fully how a continuous signal can be used to transmit discrete bits of information.

TERMS

Hamming distance: Introduced by information theorist Richard Hamming in 1950, a measure of the separation between two codewords in an error-correcting code, equal to the number of places in which they differ. For example, the English words GROVE and GRAVE, which differ in just one letter, have a Hamming distance of 1. If all the codewords of a code are separated by a Hamming distance of at least 3, then any single error can be corrected, because the resulting codeword will be “closer” to its original than to any other. With a greater minimum Hamming distance between codewords, even more errors can be corrected.

interleaving: A method of defending against bursts of errors by splitting each codeword up and distributing across a large segment of a data stream. Audio CDs use both Reed-Solomon codes and interleaving to make them very error-resistant.

redundancy: The property of language that uses many more bits (measured by the number of letters) to represent information than strictly necessary. This gives natural languages the property of error correction, but it also provides the basic vulnerability of a cipher system to cryptanalysis.

Reed-Solomon code: A powerful error-correcting code based on polynomial equations, invented by Irving Reed and Gustave Solomon. Used especially in situations where possible errors may occur in bursts, with several successive errors occurring together. Audio CDs use both Reed-Solomon codes and interleaving.

READING

Hamming, *Coding and Information Theory*, chapter 3.

QUESTIONS

- 1 Computer scientist Donald Knuth made a study of all the “word golf” chains in English. He noted that there were some words with no nearest neighbor, which he called *aloof words*. He found hundreds of these. Can you think of a five-letter aloof word?

- 2** An n -bit binary codeword has n nearest neighbors. If the code has minimum Hamming distance $d=3$, the codeword can correct any single error. That means there are $n+1$ codewords that decode to each original codeword. Because there are no more than 2^n binary codewords in all, that means the number of codewords in a $d=3$ code can be no larger than:

$$(\# \text{ codewords}) \leq \frac{2^n}{n+1}.$$

This is called the *Hamming bound*. How does it work out for the Hamming (7,4) code?

- 3** If we are transmitting 4 bits of data, there are only four possible 1-bit errors. This suggests that we might be able to get by with $\log_2(4) = 2$ additional error-correcting bits. But the (7,4) code needs 3 additional bits. Why?

Decades ago, the Voyager 1 and 2 spacecraft flew past the outer planets of our Solar System, sending back astonishing images and a vast trove of scientific data. Today, their instruments continue to make measurements of magnetic fields and the space environment. Voyager 1 is now more than 12 billion miles away from Earth, and Voyager 2 is almost as far. Their radio transmitters are not immensely powerful—about as strong as a cell phone tower on Earth—yet we continue to receive data from these spacecraft. How that is possible—and how the same principles affect communications here on Earth—is one of the most remarkable chapters in the story of information science.

Digital and Analog Information

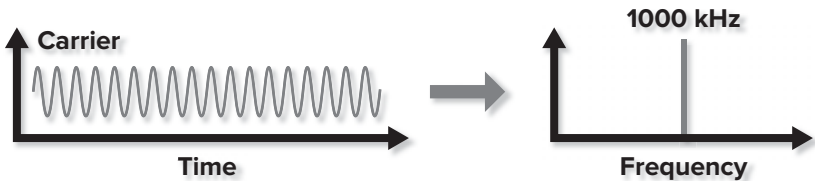
- ▶ So far, we have mostly discussed digital information, which is the information in a finite number of discrete alternatives. The 10 digits in our numbering system and the 26 letters of the alphabet express digital information. The fundamental unit of digital information is the bit, which has just two possible alternatives, 0 and 1.
- ▶ In contrast, a radio wave, such as an electrical signal in a wire or a sound wave in the air, is analog information.
 - Consider an electrical signal, a voltage. At any moment, that voltage can have any value: 1 volt, 2 volts, 3 volts, and fractional values, such as 1.4 volts and 2.718 volts.
 - In the same way, the amplitude of the electromagnetic field in a radio wave or the air pressure in a sound wave can take on a continuous range of values.

- ▶ We saw this distinction when we talked about computers. Old-fashioned analog computers, such as the differential analyzer, worked with continuous variables, including time and velocity. Computers built on Boolean circuits—on bits and logic gates—process digital information.
- ▶ The original broadcast media, ordinary radio and television, are analog information technologies. In radio, for instance, we encode the analog sound wave into the analog radio signal. There are two familiar ways of doing this: amplitude modulation (AM) and frequency modulation (FM).
 - Each of them starts out with a carrier wave, which is a simple, smooth radio signal with only one frequency. Then, this signal is modified—*modulated*—to add in the sound-wave information.
 - In AM radio, the amplitude or strength of the carrier wave is varied to represent the sound. In FM, the carrier amplitude stays constant, but its frequency is varied.
 - Both techniques have their advantages. AM is a bit simpler, while FM is more resistant to noise. That's because the external environment can readily cause fluctuations in the strength of a received signal but not its frequency.

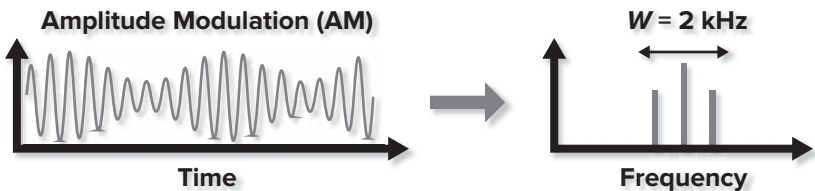
Carrier Frequency

- ▶ Both AM and FM have an effect on carrier frequency. The signal no longer has just one frequency. To understand this, we need to return to Joseph Fourier; as you recall, he showed that any signal can be regarded as a mixture of simple sine waves of different frequencies, each with a different amplitude—the Fourier transform of the signal.

- The carrier wave is all one frequency; thus, its Fourier transform is just a sharp spike at the carrier frequency. Let's suppose that frequency is 1000 kHz.

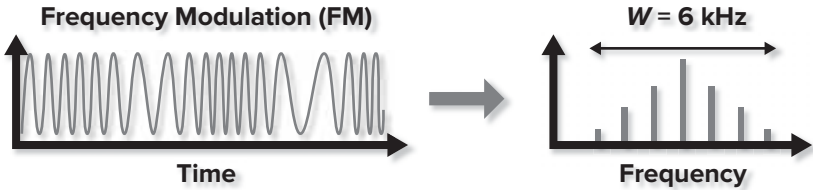


- If we use amplitude modulation to add a 1-kHz sound signal to the carrier, the resulting wave picks up two new frequencies, at 999 kHz and 1001 kHz, just 1 kHz below and above the carrier. The modulation spreads the frequency out on both sides of the carrier into regions of frequency called **sidebands**.
- The whole signal now occupies a range of frequencies. The size of that range is called the **bandwidth**, W , of the signal. Our AM wave with a 1-kHz sound signal has a bandwidth of 2 kHz.



- If we used FM to add the 1-kHz sound wave, there may be several additional frequencies in the sidebands, at 999 kHz, 998, 997 and at 1001, 1002, 1003 and so on. Because the ones farther out are much weaker, we usually worry only about the first few.

- With three important pairs of sideband frequencies, the bandwidth, W , of our signal is 6 kHz.



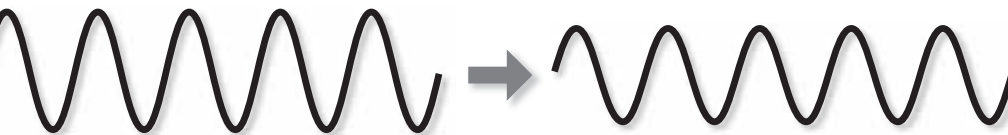
- ▶ We can tune a radio receiver to pick up only a narrow range of frequencies, just one carrier and its sidebands. Thus, even if there are many radio stations, we can listen to just one at a time, provided that the carrier frequencies are far enough apart so that the sidebands do not overlap.
- ▶ We all use the same radio spectrum, the same physical range of frequencies, and every radio signal has a bandwidth—it occupies a chunk of that “real estate.” Because we have to share bandwidth, the government makes regulations assigning different frequency ranges to different kinds of transmissions or even to particular broadcast stations.
- ▶ We can also separate signals in space. If two radio stations are far apart and their power is not unlimited, then their transmissions will not interfere with each other.
 - The best example of separating signals in space is electrical signals in wires. A single wire can carry several voltage signals separated by frequency, just like radio transmissions. In addition, if the wires are properly shielded, the signals in one wire do not interfere with those in other wires. The same is true for light signals in different fiber optic cables.
 - This means that the practical problems of “shared spectrum” are much less acute for communication over cables than for wireless communication, such as radio. With money and time, we can always add more cables if we need them, but no amount of money or time will ever add more electromagnetic spectrum.

Understanding Bandwidth

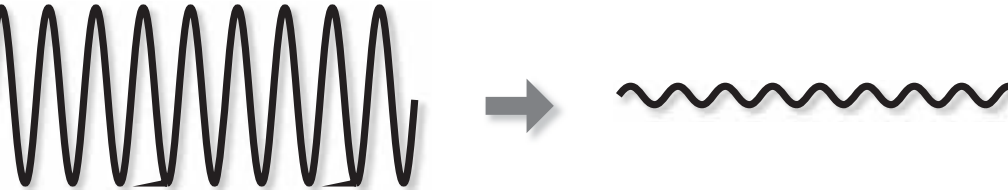
- ▶ The concept of bandwidth is important for electrical signals in wires, especially if we are using the wire to send digital information.
- ▶ To understand this, let's think about a telegraph signal. Just two voltages are used: on or off, which might be 1 volt and 0 volts. If we plot a graph of the voltage over time, a message looks like this:



- The signal is dot-dash-dash, dot-dash, dot-dot-dot-dash, dot, which spells out the word *wave* in Morse code.
 - A skilled human telegrapher could send a couple of letters per second, but a machine telegrapher could send or receive much faster. The same message could be sent in a shorter time, or several messages could share the same line. By the 1920s, most messages on long-distance telegraph lines were of this sort.
 - There is a limit, however, which is set by the bandwidth limit of the transmission line.
- ▶ Instead of just turning voltage on and off with a telegraph key, let's imagine sending a smooth sine-wave voltage at some frequency. When the frequency is low, the output signal is about the same as the input.



- When the frequency is high, the voltage is oscillating extremely fast. The electrical circuit simply does not have time to respond to the changes. The amplitude of the output signal is very small.



- Not all frequencies of signals are transmitted equally well. There is a natural upper frequency limit, determined by the physics of the circuit. In other words, there is a natural bandwidth, W , for the transmission line.
- ▶ We can think of the telegraph signal in the Fourier manner, as a combination of frequencies. But the frequencies higher than W will be lost in transmission. The received signal might look like this:



- The sudden jumps in voltage have been replaced with smoother changes, and the voltage oscillates a little around every jump, but this is still a perfectly understandable Morse signal.
- However, the more limited the bandwidth, the more the high-frequency Fourier components are quashed. The voltage changes are even less sudden.



- As the bandwidth is reduced, the distortion is worse and worse.



- Eventually, the transmitted signal has no real resemblance to Morse code. We cannot read the dots and dashes at all.



- ▶ We've been describing what happens if we send a Morse code signal through transmission lines with smaller and smaller bandwidth. But the same thing would happen if the bandwidth were constant and we tried to send the dots and dashes faster and faster. At some point, the voltage changes in the line would not be able to keep up, and the message would not get through.

The Nyquist-Shannon Sampling Theorem

- ▶ How fast is too fast? The answer to this question was discovered by an engineer named Harry Nyquist, along with Claude Shannon.
 - Nyquist posed the following problem: To send a telegraph signal, we want to transmit a series of voltage values—the on/off values of the dots and dashes. Suppose the telegraph line has a limited bandwidth, W . How many different independent voltage values can we send along it per second? In other words, how many different numbers per second can be represented by a signal of bandwidth W ?
 - Shannon posed exactly the reverse question: How many numbers per second are needed to represent a signal of bandwidth W ?

- The answer is as follows: A signal with bandwidth W (measured in Hertz, or waves per second) is exactly equivalent to $2W$ independent voltage values each second.
- ▶ Imagine a transmission line with a very low bandwidth of 100 Hz. Waves with a higher frequency than that do not get through, but lower frequencies are fine. According to Nyquist, a signal on that transmission line can describe up to 200 different voltage values per second and no more. According to Shannon, 200 voltage values per second can completely describe any signal in the line.
- ▶ That “number of voltage values” is a way of measuring the analog information in a voltage signal. If we have a signal that is T seconds long, the number of values is $2WT$. The frequency $2W$ is called the *Nyquist frequency* or *Nyquist rate*, and the whole relationship is called the *Nyquist-Shannon sampling theorem*.
- ▶ If the dots and dashes of Morse code are sent through the line more quickly than the Nyquist rate, the transmitted wave won’t contain enough different information. The signal will be undersampled.
- ▶ Closing the circle in the sampling theorem, which Shannon did in 1949, is regarded as one of his greatest achievements.
 - If we measure analog information by counting numbers—the number of independent values in a signal wave—then we face a paradox.
 - Suppose we have one number—one voltage value in a wire or one electromagnetic field value in a radio wave. That’s a continuous number; it can have any value within some range. Such a number has infinitely many decimal places. It is, in some sense, equivalent to an infinite number of bits.
 - Does that mean that one number carries an infinite amount of information? Could we, in principle, encode gigantic amounts of information in the amplitude of a single electrical pulse, radio transmission, or sound? Of course, we can’t do that.

The Effects of Noise

- ▶ As Shannon showed at the beginning of information theory, even a continuous analog signal has only a finite capacity for digital information. It can carry only a finite number of bits. The reason for this is noise, which is inescapable. Noise is always present, everywhere, though perhaps only at a very low level.
 - Consider, for instance, a piece of wire. If the wire is not attached to anything, you might expect that the voltage between the two ends of the wire would be zero. And you would be right—on average!
 - But at any given instant, the situation can be different. Because the wire contains some heat energy, the electrons are in ceaseless random motion. Sometimes it happens that there are a few more electrons near one end than the other, and that produces a momentary, very small voltage.
 - If we make extremely precise measurements, we find that the voltage in the wire is ceaselessly fluctuating, positive and negative, in a random way.
 - This was noticed in 1926 by an engineer named Bert Johnson. Shortly afterward, Nyquist deduced where the fluctuations came from. The phenomenon is now called *Johnson noise* or **Johnson-Nyquist noise**. It affects every electrical circuit. The only way to eliminate it would be to reduce the wire temperature to absolute zero, which is impossible.
- ▶ The effect of noise is to make nearby signals impossible to tell apart.
 - Consider two electrical signals with voltages of 1.000 volt and 1.001 volt—just 1 millivolt apart. These two signals represent different messages: A is represented by 1.000 volt and B by 1.001 volt.
 - We can distinguish between A and B by measuring the signal very exactly. But what if we add a little noise to the signal? Suppose the voltage is randomly changed as much as 5 millivolts, up or down. We receive a signal, measure its voltage, and get something like

1.003 volts. We have no way of knowing whether A or B was the intended message.

- ▶ If we think of an electrical signal as a point on that line—a particular value of voltage—random noise effectively smears that point out into a taller splotch or smear—a whole range of possible voltages that the signal could turn into. And if two noise splotches overlap each other very much, it is impossible to tell which received signal is which.
- ▶ If we want to prevent errors, we need to use signals that are far enough apart so that even with noise, their overlap is negligible. But spreading signals out over a wide range of voltages has a price: High-voltage signals cost energy. To use greater voltages, we will need more power for the signals. As a general fact, that power is proportional to the square of the voltage; a signal with 5 times the amplitude requires 25 times as much power.
 - P stands for the average power of a voltage signal. N stands for the power of the noise—a measure of the size of the noise splotch. Their ratio, P/N , is the well-known **signal-to-noise ratio**, sometimes abbreviated SNR.
 - By making a simple assumption about the noise—the shape of the splotch—Shannon was able to calculate the information capacity, C , of one noisy analog signal. In fact,

$$C = \frac{1}{2} \log_2 \left(1 + \frac{P}{N} \right).$$

That's the number of bits that the signal can convey.

- As long as the noise isn't zero and we don't use an infinite amount of power, the capacity is finite. It depends on the SNR.
- ▶ The SNR is often expressed in a logarithmic way, using **decibels** (dB). A given number of decibels expresses a given ratio, using base-10 logarithms: 0.0 decibels means a ratio of 1; 10.0 decibels means a ratio of 10; 20 decibels means a ratio of 100. Each time the ratio multiplies by a factor of 10, the number of decibels increases by 10.

- Suppose your computer has a wireless Internet connection. If you have a very strong connection to the local access point, your SNR might be around 1000, or 30 decibels. Shannon's formula gives a capacity of around 5 bits per transmitted signal.
- Sometimes, however, your connection is not as good; the signal is weaker, or the ambient radio noise level is worse. Then, your SNR might only be a ratio of 30, or 15 decibels. That allows only 2.5 bits per transmitted signal.
- Wireless networks are designed to adapt when the SNR ratio is low. The link still works, but it takes longer to send or receive a given amount of information.

Combining Limitations on Analog Signals

- ▶ We've now discussed two limitations on analog signals: One is due to the bandwidth, the range of frequencies allowed in the signal; the other is due to noise and the limited amount of power we can put into the signal. We can now combine the results.
- ▶ A signal wave with limited bandwidth W is equivalent to sending $2W$ signal values per second. For a given SNR, each of those signal values can carry only a finite number of bits. From these two ingredients, we can find the maximum information rate in bits per second.
- ▶ Because this is information capacity, we will call it C , but we will write it in cursive to remind us that this is measured in bits per second (an information rate), rather than just bits. The formula, called the *Shannon-Hartley formula*, is as follows:

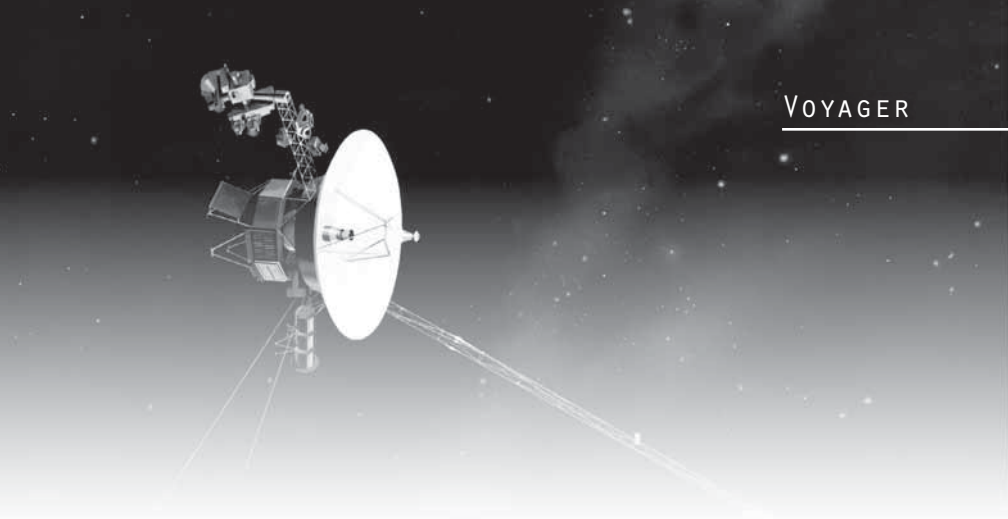
$$\mathcal{C} = W \log_2 \left(1 + \frac{P}{N} \right).$$

The Shannon-Hartley formula is the basic law that governs how much information can be sent over a noisy transmission line or radio link.

- ▶ The formula is a little trickier than it appears. For instance, at first glance, it seems that we can always double the capacity C by doubling the bandwidth W . However, when we open up our communication system to a wider range of frequencies, we also open it up to more noise. The noise power also depends on the bandwidth.
- ▶ Let's work out a simple example. We previously considered a transmission line with bandwidth $W=100$ Hz. Let's suppose the SNR is 2. Then, the Shannon-Hartley formula tells us that the maximum information rate we can send is 158 bits per second.
 - Now suppose we double the bandwidth to 200 Hz. This most likely doubles the power of the noise, as well. With the same signal power, we'll wind up with an SNR of only 1. Then, the information rate changes to 200 bits per second.
 - As we make our bandwidth wider and wider, we let more noise in, and that limits the information capacity at a given power. Even a bandwidth of 10,000 Hz gives us less than 288 bits per second.
 - When the signal-to-noise ratio is 1, the maximum information rate equals the bandwidth. In our example, 200 Hz becomes 200 bits per second.
 - The Shannon-Hartley formula tells us that the information rate depends on both bandwidth and SNR.

The Spacecraft Voyager

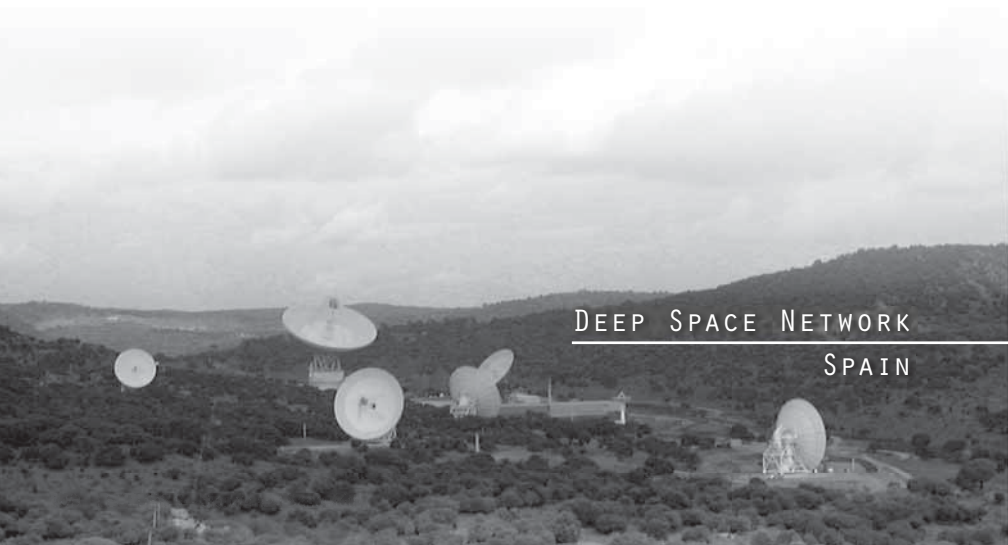
- ▶ Let's go back to where we began, with the radio data link from the Voyager spacecraft, 20 billion kilometers from Earth. The Voyager radio transmitter has a power output of only around 20 watts.
- ▶ The power that matters, of course, is the actual power that reaches Earth. If Voyager broadcast those 20 watts equally in all directions, the radio signal on Earth would be incredibly weak. Each square meter would



VOYAGER

receive only 4×10^{-27} watts—far too small to detect. Voyager, however, aims its signal very precisely at Earth, and the received radio signal here contains 2.6×10^{-22} watts in each square meter—still quite small.

- To gather as much radio power as possible, NASA uses the giant parabolic antennas of the Deep Space Network, located in California, Spain, and Australia. Each one is 70 meters in diameter, with an area of almost 4000 square meters. The total signal power (P) received from Voyager is about 10^{-18} watts.



DEEP SPACE NETWORK

SPAIN

- ▶ To have any hope of working, the system must somehow reduce the noise power (N) to extremely low levels. There are both internal sources of noise (produced by the radio receiver itself) and external sources (including terrestrial radio transmissions and noise from deep space). The antennas of the Deep Space Network are extremely high-**gain** antennas; thus, they accept only radio energy from a narrow range of directions in the sky. That reduces the received noise.
- ▶ Even so, the Deep Space Network is just barely able to receive the Voyager data. And as the two spacecraft get farther away, those radio signals diminish. The SNR decreases, which in turn, reduces the possible information rate.
 - When the spacecraft visited Jupiter, less than 1 billion kilometers from Earth, data was returned at 115,000 bits per second. At Saturn, the rate was less than half that.
 - Every spacecraft to the far reaches of the Solar System faces the same challenge. The New Horizons probe, which flew by Pluto in 2015, transmits its data at only 1200 bits per second.
 - Now, Voyager 1 is more than five times farther away than Pluto. Its data can be transmitted at only 160 bits per second, almost 800 times more slowly.
 - Voyager is not designed to transmit at any slower bit rate. Thus, at some point in the next decade, Voyager 1 will simply be too far away for us to hear its messages; though for a few years longer, radio telescopes will be able to find its carrier wave, shining faintly in the radio sky like a star.
- ▶ The deep space communication system that began with Voyager is an amazing engineering achievement. It is testimony to our ability to shape information into forms that can be conveyed almost inconceivable distances and understood amidst a cacophony of competing signals.

- The fastest earthbound computer networks communicate just about a billion times faster than Voyager: 100s of gigabits per second. The total information traffic on the Internet is perhaps 1000 times greater still.
- Yet these extremes are governed by the same laws of bandwidth and SNR that are at work all around us, in the communication systems we use every day.

TERMS

bandwidth: The range of frequencies used by an analog signal, be it sound, radio, or an electrical voltage. Every communication channel has a limited bandwidth, a finite range of frequencies that can be conveyed faithfully. Along with the **signal-to-noise ratio**, the bandwidth limits the information rate of an analog signal, according to the Shannon-Hartley formula.

decibel: A logarithmic measure of the ratio of two magnitudes, often used to describe the signal-to-noise ratio (SNR). Each decibel difference of 10 represents a factor of 10 in power. Thus, if signal A is 30 decibels more than signal B, it has 1000 times the power. The decibel scale is often used to measure the loudness of a sound relative to the quietest sound that a normal human being can hear.

gain: A measure of the geometric advantage of a directional antenna: the ratio of the directed signal power to the power of a signal that is spread equally in all directions. (A complementary definition describes the gain of a receiving antenna.) A parabolic high-gain antenna can achieve gain values of thousands or millions of times.

Johnson-Nyquist noise: Random electrical noise in a circuit produced by the thermal motion of electrons in it. Discovered by Bert Johnson and Harry Nyquist at Bell Labs in 1926. Extremely sensitive radio receivers, such as those used by radio telescopes, are often cooled to a low temperature to reduce this internal source of interference.

sidebands: The spread of radio energy to frequencies on either side of the basic carrier frequency, produced when information is added in the form of AM or FM radio signals (amplitude or frequency modulation). The presence of sidebands determines the bandwidth of the radio signal.

signal-to-noise ratio: Also known as SNR, the ratio of the power of a signal to the power of the interfering noise. This is often described logarithmically, using decibels. Along with the signal bandwidth, the signal-to-noise ratio limits the information rate of an analog signal, according to the Hartley-Shannon entropy.

READINGS

Doody, *Basics of Space Flight*, chapters 10 and 18.

Pierce, *An Introduction to Information Theory*, chapters 9–10.

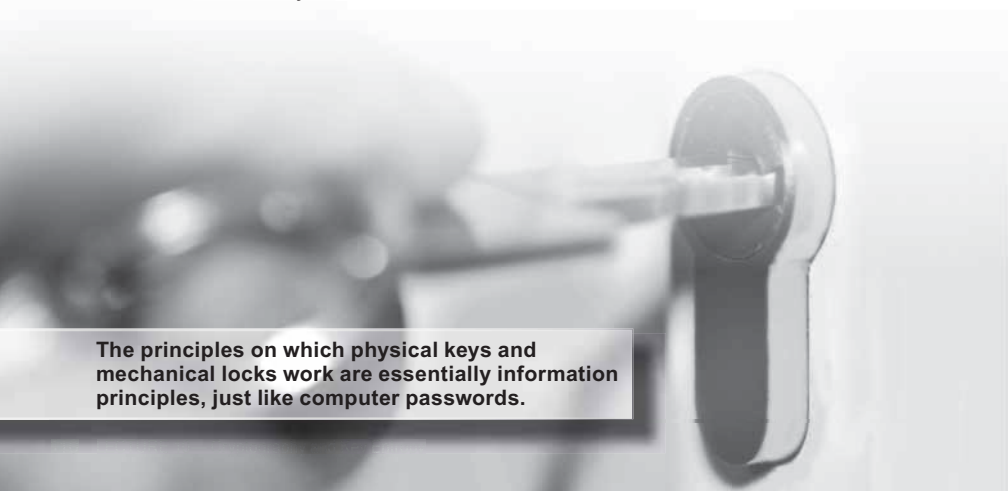
QUESTIONS

- 1 To get insight into the Nyquist-Shannon sampling theorem, suppose we sample a signal at 200 samples per second. As it happens, all of our sample values are zero. Can we conclude that the whole signal is zero? What is the lowest-frequency sine-wave signal that could spoof us by making our samples zero, even though the signal is not?
- 2 We are trying to improve a signaling system to increase its maximum information rate C . Is it better to (a) double the bandwidth W without changing the signal-to-noise ratio P/N , or (b) double the signal-to-noise ratio P/N without changing the bandwidth? You might wish to try out some calculator examples.

Thus far, we have focused on communication—how we can use codes to communicate information efficiently and to defend information from errors in transmission. These are the questions addressed by Shannon’s first and second fundamental theorems. But codes can have another purpose, as well: to conceal information, to protect the privacy of messages we transmit. That is the concern of one of the main branches of the science of information, the field of cryptography.

Defining *Cryptography*

- ▶ Cryptography encompasses all those situations in which the usefulness of information depends on its secrecy. Passwords are the simplest example. The point of a password is that the right people know it and outsiders do not. In effect, a password is a kind of **key**, and only people who possess the key can open the lock and be granted access.
 - The strength of that lock is determined by how many possible keys there are, which in turn is measured by the entropy of the password. If the entropy is low, there are relatively few possible passwords, and a determined adversary might be able to guess the right one in a relatively short time.



The principles on which physical keys and mechanical locks work are essentially information principles, just like computer passwords.

- With a password chosen from a gigantic set of possibilities, the entropy is high, and we can be confident that the adversary will not be able to guess the password.
- ▶ The cryptography of secret communication is tremendously important. Internet commerce, for example, depends on our ability to send sensitive financial data over communication networks that we do not completely control and cannot completely trust. We rely on cryptography to keep that information safe.
- ▶ Cryptography mostly considers codes of a special type, called *ciphers*. We begin with a message in some ordinary, readily understood form, such as English text. This is called the **plaintext**. A cipher is a mathematical rule by which the plaintext is converted to a form that outsiders cannot easily read. This form of the message is called the **ciphertext**. The process of transforming the plaintext into the ciphertext is called encryption or *encipherment*. Converting the ciphertext back to plaintext is called *decryption* or *decipherment*.

Cipher Security

- ▶ One of the earliest ciphers we know about was the *Caesar cipher*, so called because it was used by Julius Caesar for his private correspondence. In the Caesar cipher, the letters of the alphabet are written in a circle. Then, each letter of the plaintext message is replaced with the letter that is three places counterclockwise on the circle.
- ▶ To decide how secure a cipher is, we must make an assumption about how much the eavesdropper knows. The standard assumption in cryptography is called **Kerckhoffs's principle**, after the 19th-century cryptographer Auguste Kerckhoffs, who first recommended it.
- ▶ To be on the safe side, Kerckhoffs said, we should assume that the eavesdropper knows in a general way what type of cipher is being used. A third party, Eve, for instance, might know that Alice and Bob are using

a Caesar shift cipher of some kind but not how many places the letters are shifted.

- ▶ Shannon rediscovered Kerckhoffs's principle and gave it an especially pithy form that we now call **Shannon's maxim**: "The enemy knows the system." In Shannon's terms, a cryptographic system is a set of possible codes that Alice and Bob might use. Eve does not know which code they are using at the moment, but we must assume that she knows what the possibilities are.
- ▶ The information that Eve does not have—the choice of code within the system—is called the *key* to the code. A cryptographic system is secure if the entropy of the key is high enough so that Eve would have a hard time guessing it.
- ▶ If our cryptographic system is a Caesar shift cipher, the key is the amount by which we shift each letter. We can simply note how the plaintext letter A is to be encrypted, and all of the other letters follow from that. For example, Caesar's original cipher has the key X, because A is shifted to X. In this type of system, there are only 26 possible keys, and the entropy of the key is low: $\log_2(26) = 4.7$ bits.
- ▶ A Caesar shift cipher is a type of **substitution cipher**, in which letters are replaced by other letters (or other symbols) according to a constant rule. But there are many more ways to do that than a simple shift around the alphabet circle. We can represent this more general sort of code by a table, with the plaintext letters along the top and the corresponding ciphertext letters below.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
M	B	E	A	O	R	L	N	S	I	Z	C	K	Q	W	X	U	V	T	J	F	H	G	D	P	Y

- The bottom row is a reordering (or *permutation*) of the top row. How many permutations of 26 letters are there? As we fill in the bottom row, we have 26 possible choices for the first letter, 25 choices for the second, 24 choices for the third, and so on. The total number

of possible permutations is: $26 \times 25 \times 24 \dots$. This number is roughly 400 trillion trillion; that's the number of possible substitution ciphers.

- In this system, the entropy of the key—the amount of key information—is, therefore, $\log_2(26!) = 88$ bits. Thus, a general substitution cipher is much more secure than a Caesar shift.

Cryptanalysis

- ▶ Cryptography tells us that even a substitution cipher is not all that secure. This relates to the fact that cryptography is not only about using codes to keep your own messages private; it is also about figuring out the codes that other people are using to conceal their secrets. This is called **cryptanalysis**. Let's work through an example.

- ▶ We begin with a ciphertext:

[SV XVL PWHWMK VJP NWQPWL LV LCW WXWZB].

- ▶ We know that the plaintext is an English phrase or sentence and that the code used is one of the 400 trillion trillion possible substitution ciphers.
- ▶ Our first step to break the code is to examine the frequencies of letters in the ciphertext. The most common letter is W, which occurs seven times. This probably means that W is the letter E, which is the most common letter in English, giving us:

[** ** *E*E** ** *E**E* ** **E E*E**].

- ▶ The next two most common ciphertext letters are V and L, which each occur four times. And the next most common letters in English are T, A, and O. Looking at the message, we can hypothesize that ciphertext L stands for T. That conclusion comes from the three-letter group LCW, which could be T-H-E, the most common English word. We'll tentatively add the T:

[** **T *E*E** ** *E**ET T* THE E*E**].

- ▶ We next note a two-letter group, LV, which is almost certainly T-O. That would mean that ciphertext V represents O, which fits in with the letter frequencies. It also helps to confirm our previous guess about T. We now have:

[*O *OT *E*** O** *E**ET TO THE E***].

- ▶ Continuing in this way, we arrive at the message:

[DO NOT REVEAL OUR SECRET TO THE ENEMY].

- ▶ In this sort of problem, we proceed by a series of guesses, guided by the frequencies of the letters and the combinations of letters that make words. This basic technique is called **frequency analysis** and has been used for some time. Because it depends on the statistics of the plaintext, it works only for longer messages.

The Vigenère Cipher

- ▶ Because substitution ciphers are vulnerable to frequency analysis, at least for long messages, cryptographers have spent centuries finding ways to make better, less vulnerable ciphers. One of the most significant of these was the **Vigenère cipher**, named after the 16th-century French diplomat Blaise de Vigenère.
- ▶ The Vigenère cipher is straightforward to use, and it is much harder to break than a simple substitution cipher. Indeed, it became known as *le chiffre indechiffable*, the “indecipherable cipher.” It was used by the Confederate Army during the American Civil War.
 - In the war, both sides used cipher disks, which would have been carried by signal officers. Such a disk has two wheels with alphabet circles. The inner wheel (which is fixed) corresponds to the plaintext of the message, and the outer wheel (which rotates) corresponds to the ciphertext.

- With a cipher disk, the plaintext letters can be easily lined up with the corresponding ciphertext letters by simply rotating the outer wheel.
- ▶ A Caesar cipher has only a single letter as its key, but the key to a Vigenère cipher is a whole word or phrase—a sequence of letters. To illustrate, let's suppose the key word is VICTORY. We write the key word over and over on one line, and underneath it, we write the letters of our plaintext message:

[VICTORYVICTORYVICTORY...]
[ATTACK AT DAWN]

- ▶ Now we use the cipher disk to encipher each letter in the plaintext, using the key phrase to decide how to shift the wheel. The first letter uses the V shift, so A is lined up with V, and the plaintext A becomes ciphertext V. The second letter uses the I shift, so A is lined up with I, and the plaintext T becomes ciphertext B. Continuing, we get:

[VBVTQBYOLCPB].

- ▶ For convenience in transmission, we can break up the ciphertext in groups of four: [VBVT QBYO LCPG]. That's the message we send in Morse code over the telegraph.
- ▶ Because the Vigenère cipher uses a phrase of many letters for its key, it has many more possible keys than a Caesar cipher. A higher-key entropy makes the code more secure. Our example was the seven-letter word VICTORY, but we could choose a key as long as we like.
- ▶ Frequency analysis is hopeless for breaking the Vigenère cipher. The same letter is enciphered in several ways, depending on where it appears in the message. In this way, the statistical regularities of English text are hidden by the continually shifting cipher. Charles Babbage, however, devised a technique for breaking *le chiffre indechiffirable*. Though Babbage did not publish his method, it was rediscovered later and published by the German Friedrich Kasiski.

- To use the Babbage-Kasiski method, we need a long ciphertext, much longer than the key phrase. We then search for short sequences of letters that reoccur and might be associated with common short words in English, such as THE and AND, or common letter groups, such as TION.
- Our cryptanalysis then proceeds by a series of plausible guesses. Incorrect guesses quickly lead to dead ends, while correct guesses open up further possibilities.
- The long struggle between encryption and cryptanalysis—between concealing secrets and uncovering them—poses a fundamental question for the science of information: Is it true that any cipher can be broken by a clever enough analysis, or might we find a true *chiffre indechiffable*—a secret code that is perfectly secure? We will begin to answer that question in the next lecture.

TERMS

ciphertext: In a cipher system, the coded message that is actually transmitted and could be intercepted by an eavesdropper. Without the key, however, the plaintext remains secret.

cryptanalysis: The art and science of penetrating other people's secret codes, deducing their plaintext messages from intercepted ciphertexts. Shannon's information-theoretic approach to cryptography showed when cryptanalysis is possible in principle, but the practicality of the task often depends more on its computational difficulty.

frequency analysis: A technique, first described in the 9th century by the Arab philosopher Al-Kindi, for breaking a substitution cipher by analyzing the frequency with which various letters appear.

Kerckhoffs's principle: The practical rule, first formulated by Auguste Kerckhoffs, that the designer of a cryptographic system should assume that any adversary knows in a general way the type of system that is used. Only the key information is secret. Later, Claude Shannon reformulated this rule as: "The enemy knows the system."

key (cryptographic): In a cipher system, shared information between sender and receiver that is not shared by the eavesdropper. The secrecy of the key guarantees the privacy of the communication. One measure of the security of the system is the entropy of the key; the larger the entropy, the more secure the system.

plaintext: In a cipher system, the uncoded "plain language" message whose secrecy is to be protected.

Shannon's maxim: See **Kerckhoffs's principle**.

substitution cipher: A cipher system in which each letter of the plaintext is replaced by another letter or symbol according to a constant rule. Substitution ciphers are relatively easy to break by frequency analysis, if the ciphertext is long enough.

Vigenère cipher: A cipher system using a continually shifting set of substitution ciphers, determined by a repeated key phrase of any fixed length. Known as *le chiffre indéchiffrable* ("the indecipherable cipher") because it cannot be broken by simple frequency analysis, but Charles Babbage invented a way to break it.

READING

Singh, *The Code Book*, chapters 1–3.

QUESTIONS

- 1 It is an amusing exercise to try to discover English words that become other English words when translated into ROT 13. Find some.
- 2 Cryptography that ignores Kerckhoffs's principle (or Shannon's maxim) relies on potential eavesdroppers not knowing what sort of system is being used. This is sometimes called *security by obscurity*. Can you think of an example? Comment on its security.

Cryptanalysis and Unraveling the Enigma

LECTURE

11

Encryption—the process of protecting secrets—and cryptanalysis—the process of uncovering secrets—are two sides of cryptography that are continually at war. As one side becomes more sophisticated, the other becomes more difficult, driving innovations on both sides. In fact, the whole history of cryptography is a kind of arms race between encryption and cryptanalysis. In the first half of the 20th century, some of the greatest triumphs of cryptanalysis were realized. Code-breaking changed the course of history more than once and opened new vistas of history to our eyes. This was also the era when Claude Shannon used information theory to show what secrecy actually meant and how and when it could be broken.

The Enigma System

- ▶ In the aftermath of World War I, the German military adopted the most advanced cipher system available at the time: the **Enigma system**. Enigma was a cipher machine, originally designed for commercial use. With a few modifications, it became the standard cryptosystem for the German military. Enigma was the primary German code that the Allied cryptanalysts faced during World War II.
- ▶ There were many Enigma variations, but the typical setup was as follows: The machine had a keyboard of 26 letters. There was also a light board with 26 bulbs, one for each letter. An electrical signal from the keyboard to the light board ran through three rotor wheels, each of which had 26 positions and contained circuitry to scramble the electrical connections in a different way. The rotors came in sets of five for each machine and could be switched out or changed around. At the front of the case, there was a switchboard-type plug arrangement called the *steckerboard*. Using cables, the operator could connect up to 10 pairs of letters.

- ▶ An Enigma key consisted of three parts. First, there was a choice of three rotor wheels out of the set of five provided with the machine (60 possibilities in all). Second, each rotor was given an initial setting, labeled by a letter for each rotor ($26^3 = 17,576$ settings). Finally, the 10 cables of the steckerboard could be connected in 150 trillion possible ways. These three factors yielded a vast number of possible Enigma keys—about 160 million trillion possible key combinations.
- ▶ To use the Enigma machine, the operator hit a key for the plaintext. The bulb lit up, giving the ciphertext for that letter. The rotor then advanced, so that the next letter would be enciphered with a different rotor configuration.
- ▶ The Enigma was an amazingly sophisticated cipher system, but even as the German military began to adopt it during the 1930s, the Polish Cipher Bureau was already able to break it, at least occasionally. In 1939, the Germans invaded Poland, and the Second World War began. Many of the Polish cryptographers escaped to England, bringing with them their hard-won discoveries about Enigma. The British then launched their own cryptanalytic effort, led by the brilliant Cambridge mathematician Alan Turing.
- ▶ The British attack on Enigma was based on two elements: a careful mathematical analysis of the Enigma system that revealed a few slight weaknesses and the development of a special-purpose computer called a *bombe* that automated some of the work.

Breaking the Enigma

- ▶ Different variations in the Enigma system required the use of different techniques for breaking it, but we can identify a couple of important steps in the main effort.
- ▶ First, the Enigma system exchanged letters. With a given setting, if A was transformed to X, then X would be transformed to A. That meant that encryption and decryption were the same procedure. But it also

meant that no letter was ever encrypted as itself. A might be encrypted as D, J, or W but never as A. That provided a tiny clue. A letter in the ciphertext revealed a fraction of information about the plaintext.

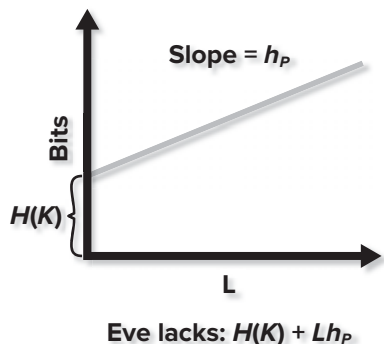
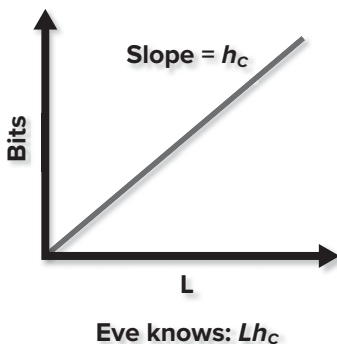
- ▶ In addition, when a message was intercepted, there was some context for it. The receivers knew something about who was sending the message, who it was intended for, or the overall situation in which it was transmitted. There might have been good reason to believe that a particular word or phrase, such as SUBMARINE, might appear in the message. In cryptography jargon, a possible piece of plaintext that can be used to help in cryptanalysis is called a *crib*.
 - To find out where the word SUBMARINE might appear, an operator could slide that word along the ciphertext. Often, one of the letters in SUBMARINE would match a letter in the ciphertext. Knowing that in Enigma, no letter is encoded as itself, the operator would then know that SUBMARINE did not appear at that location.
 - If the crib was long enough, there would be only a few possible places where it could occur in the message.
- ▶ The next step was one of Turing's great discoveries about Enigma. Suppose we have a crib in a possible location within the ciphertext. It can happen that the plaintext and ciphertext form a loop. The plaintext letter A is encoded as R. The next place over, plaintext R is encoded as N. Two places down, plaintext N is encoded as A. A to R to N and back to A makes a kind of closed loop. Such loops occur surprisingly often.
 - Turing realized that this loop structure did not depend on the plug settings of the *steckerboard* but only on the rotor settings.
 - By concentrating on loops, the Enigma problem could be separated into two pieces: the rotor settings and the steckerboard. Bombes then enabled the operator to try all possible rotor settings. Each setting that made the right kind of loop was a possible piece of the Enigma key.

- ▶ Through most of the war, British cryptanalysts were able to decrypt Enigma messages on the same day they were transmitted, and this real-time military intelligence was of immense consequence in the war. Some say that it may have meant the difference between victory and defeat.

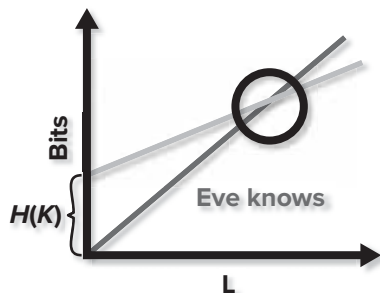
Information Theory and Cryptography

- ▶ During World War II, Shannon was working out his ideas about codes and communication and realized that information theory could provide a new approach to cryptography. He tackled the subject by asking some fundamental questions: What is a cipher? What makes it secure? What does it mean to break a cipher? When is that possible?
- ▶ Shannon's analysis began with entropy. If we have M possible messages, we have an entropy $H = \log_2(M)$, measured in bits. That's the amount of information we lack about the message. Lacking H bits, the number of possible messages, roughly speaking, is $M = 2^H$.
- ▶ The more we learn about the message, the smaller our missing information, H . There are fewer possible messages consistent with what we know. Eventually, we arrive at the point where we know everything, and there is only one possible message: $H = 0$ and $M = 1$. Then, we know what the message is, or to be more precise, we know everything that is required for determining what the message is. We might still need to do a great deal of calculation to turn the message into an understandable form, but we do not need to receive any more bits.
- ▶ Suppose Alice is sending a secret message to Bob, while Eve listens in. We'll call the plaintext P and the key K . The ciphertext, which Eve might intercept, is C .
 - Now imagine that Alice and Bob use a substitution cipher. Eve intercepts a short ciphertext message: WBJKRTQ.

- Eve literally cannot tell what the message means. All she knows is that it is seven letters long and uses no letter twice. Aside from that, it could mean anything, depending on the key. Eve's set of M possible plaintext messages is vast.
- ▶ But a substitution cipher can be broken by frequency analysis if the ciphertext is long enough. Shannon tells us the following:
 - Before any message is sent, Eve starts out with no information. When she sees the ciphertext, the amount of information she gains is $H(C)$, the entropy of C . Each ciphertext letter adds $\log_2(26) = 4.7$ bits of data for Eve. We'll designate that value as h_C .
 - If the message is L letters long, the ciphertext information $H(C) = L \times h_C$. That's how much information Eve gets.
- ▶ What information does Eve lack? From the beginning, Eve does not know the key, K . Thus, she starts out with missing information $H(K)$, the key entropy. Once the message is sent, she also lacks $H(P)$, the information content of the plaintext. Each plaintext letter adds h_P bits of entropy; thus, a message of length L has $H(P) = L \times h_P$. The total amount of information Eve lacks is $H(K) + L \times h_P$.
- ▶ We can graph both the amount of information Eve has and the amount she lacks, as shown below:



- Each horizontal axis is the length, L , of the message. Each vertical axis is bits of information.
 - The graph on the left shows the amount of information Eve has. It starts at zero and gets larger as the message gets longer. The slope of the line is $h_C = 4.7$ bits per letter, the entropy of one letter of ciphertext.
 - The graph on the right shows the amount of information Eve lacks. It starts at $H(K)$, the key entropy, and gets larger as the message gets longer. The slope of the line is h_p , the entropy of one letter of plaintext.
- The value of h_p is not 4.7 bits per letter. As we have seen, English is redundant. Our experiments led us to estimate that the entropy of English text is less than 2 bits per letter. Shannon's estimate was about 1.5 bits per letter. Thus, our 26-letter alphabet, with 4.7 bits per letter, introduces a great deal of redundancy. For each letter, $4.7 \text{ bits} - 1.5 \text{ bits} = 3.2 \text{ bits}$ of redundant information, on average. The per-letter redundancy of English is designated D , and $D = h_C - h_p$, or 3.2 bits per letter.
- As mentioned earlier, redundancy gives human languages a natural error-correcting ability, but it also has cryptographic implications. It means that the graph of Eve's missing information goes up at a shallower slope, only about 1.5 bits per letter. If we place the two lines on the same graph, we notice that they eventually intersect.
- Of course, Eve can never learn more than there is to know about the message. The ciphertext line can never go above the other one. When the two lines get close to each other, the ciphertext line bends over, and they run together.

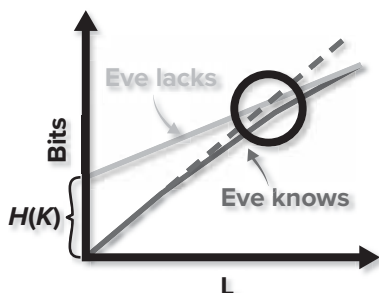


- But here is the point:
The difference becomes negligible.

- The difference is the net amount of information that Eve lacks: $H(K) + H(P) - H(C)$. That quantity shrinks to zero eventually.

When that happens, there is only one possible

key and one possible plaintext consistent with the ciphertext that Eve has. That means she can break the code.



- When does this happen? It happens near the point of intersection of the two straight-line graphs. Where does this happen? Let's solve for L .

- At the intersection point, $L \times h_C = H(K) + L \times h_P$, or $H(K) = L \times (h_C - h_P)$, which is $L \times D$.

- Eve's information catches up when the number of letters equals the key entropy divided by the redundancy per letter: $L = H(K)/D$.

- Shannon called this the **unicity distance**, which suggests a sort of "one-ness." If the message is much longer than the length, L , there is only one key, K , and plaintext, P , consistent with the ciphertext, C . The ciphertext alone provides enough information for Eve to break the code.

- If Alice and Bob use a substitution cipher, there are $26! = 400$ trillion trillion possible keys. That's a key entropy of $H(K) = 88$ bits. Using Shannon's estimated English redundancy, $D = 3.2$ bits per letter, the unicity distance is $88 \text{ bits} / 3.2 \text{ bits per letter} = 28$ letters. A ciphertext much shorter than this is not enough to break the code; many possible plaintexts will be consistent with the ciphertext. But if the ciphertext is much longer than 28 letters, there is probably only one sensible plaintext consistent with it, and it ought to be possible to break the code.

- ▶ The unicity distance is not an exact line. It is an approximate rule of thumb. But it is an excellent way to estimate how much data is needed to break a code.

The Unicity Distance for Enigma

- ▶ The unicity distance tells us only when a solution is possible—when a code can, in principle, be broken. It does not tell us what we will have to do to break it. To understand this idea, let's estimate the unicity distance for Enigma.
- ▶ First, we must determine the key entropy, $H(K)$, which in this case is 67 bits. That's actually less than the key entropy of a substitution cipher, and the unicity distance is correspondingly less: $L = (67 \text{ bits}) / (3.2 \text{ bits per letter}) = 21 \text{ letters}$.
- ▶ In principle, an Enigma ciphertext message much longer than about 21 letters has only one reasonable solution. In theory, we need less material to find the solution to an Enigma message than a substitution cipher, but in practice, Enigma is a much more difficult problem.
 - As we've seen, we can solve a substitution cipher in steps, starting with frequency analysis. The substitution cipher represents a high entropy mountain—88 bits high—but we can climb it in easy stages.
 - Enigma's entropy mountain is not quite so high, but the sides appear to be sheer cliffs. With his discovery that the rotor and steckerboard settings of Enigma could be solved separately, Turing managed to scale a much lower cliff, then follow an easy step-by-step ascent the rest of the way.

The Decipherment of Linear B

- ▶ Not all cryptanalysis has to do with espionage and war. In fact, one of the great achievements of cryptanalysis just after World War II was the decipherment of Linear B by Michael Ventris and John Chadwick.

- ▶ Linear B was a system of writing used on Crete and elsewhere in the eastern Mediterranean about 3200 years ago. Many of the examples of it we have are inscribed on clay tablets, and many of these appear to be inventories, illustrated with small pictures of chariot wheels, arrows, and so on. With the pictures, we can guess at some of the written symbols, but the rest are mysterious.
- ▶ The man who deciphered Linear B was not a professional archaeologist or a linguist but an English architect named Michael Ventris. Long fascinated by the Linear B problem, after World War II, Ventris began to make a serious effort to decipher the code. In this effort, he had a few clues, including the fact that the number of symbols suggested a syllabic rather than an alphabetic system.
- ▶ Both Ventris and an American scholar, Alice Kober, noticed some fascinating regularities in Linear B. These regularities were signs of the redundancy of the underlying language, and as we know, redundancy is the information basis for cryptanalysis. However, without knowing what the underlying language might be, it was hard to see how to proceed.
- ▶ Ventris began to search the inscriptions for place names, which are often very old. The place names that Ventris conjectured gave him a few syllables to go on, for instance, “ko-no-so” for Knossos, the place where many of the tablets were found. These place names gave Ventris a kind of crib to Linear B.
- ▶ Gradually, Ventris was forced to consider a surprising hypothesis: The language of Linear B looked like Greek, although a previously unknown and much older form of the language. In fact, it resembled the archaic form of Greek that had been recently reconstructed from linguistic evidence by the Cambridge scholar John Chadwick. Working together, Ventris and Chadwick were able to establish beyond a doubt that the Linear B inscriptions were indeed written in this archaic Greek.

TERMS

Enigma system: Sophisticated encryption system used throughout the German military during World War II.

unicity distance: The amount of ciphertext data required to uniquely specify the original plaintext. For a simple substitution cipher, Shannon estimated a unicity distance of 28 letters. Thus, any enciphered message much longer than this should be breakable by cryptanalysis.

READINGS

Hinsley and Stripp, eds., *Codebreakers*, especially part II.

Shannon, "Communication Theory of Secrecy Systems," pages.cs.wisc.edu/~rist/642-spring-2014/shannon-secrecy.pdf.

Singh, *The Code Book*, chapters 4–5.

QUESTIONS

- 1 The Enigma system and ROT 13 are vastly different ciphers, yet they share one essential property. What is it?
- 2 A particular Vigenère cipher is known to have a random key phrase 20 letters long. Calculate Shannon's estimate of the unicity distance of the cipher. Does this seem reasonable?
- 3 Can data compression make a cipher harder to break?

Unbreakable Codes and Public Keys

In 1943, during World War II, Claude Shannon was called on to evaluate the security of the encrypted telephone link that allowed Franklin Roosevelt and Winston Churchill to confer in secret across the Atlantic Ocean. Codenamed SIGSALY, this system was the first ever designed to protect the privacy of that kind of electronic communication. In this lecture, we'll look at how this amazing technological achievement worked. We'll then turn to some of the problems addressed in post-Cold War cryptography.

SIGSALY

- **SIGSALY** was the first telephone system to convert analog voice signals into digital information, using a technique called *pulse code modulation*. The digital signal amounted to a highly compressed version of the original voice data. As a result, the compressed voice data had much less redundancy than the original. As we've seen, cryptanalysis relies on that redundancy, and reducing it makes the eavesdropper's job more difficult.



SIGSALY APPARATUS

- ▶ The real secrecy of SIGSALY came from the cryptographic key: a pair of identical phonograph records, one at the transmitting station and one at the receiving station. Each record contained the exact same random electronic noise. Before the SIGSALY voice data was transmitted, it was added to the random noise from the record. At the other end, the same noise was subtracted from the received signal, yielding the original. That, in turn, was decompressed and turned into a recognizable, understandable telephone voice.
- ▶ Shannon concluded that the SIGSALY system was indeed secure.
 - The enemy can intercept only what is actually transmitted: the combined signal of voice data plus random noise (the “ciphertext”). But the enemy knows nothing about the contents of the noise record (the “key”).
 - Therefore, any voice transmission whatsoever is consistent with what the enemy learns. The phone conversation might be about military plans, weather reports, or baseball results. For any plaintext voice data, there exists a possible key (a noise record) that would make it look just like the received ciphertext signal. Thus, the eavesdropping enemy learns nothing.
- ▶ According to Shannon’s description of cryptanalysis, secrecy comes from the gap between the secret information and the information available to the enemy.
 - That gap starts out with the entropy of the key, $H(K)$. But because of redundancy, D , in the plaintext message, the gap narrows over time. When the gap closes to zero, the enemy has enough information to read the message, at least in principle. The unicity distance, $L = H(K)/D$, tells us how long a message is required before the code can be broken.
 - Notice the assumption in this analysis. The secret key has a fixed entropy, $H(K)$. Therefore, the secrecy gap starts out at a fixed size and gets narrower over time. That’s how the enemy catches up.

- But in SIGSALY, the key information grows along with the message—more noise is added from the record. And each conversation uses new records, with new random noise. The secrecy gap grows wider and wider. The unicity distance is infinite.
- ▶ This general approach to cryptography was known in Shannon’s time as a **one-time pad**.
 - Picture Alice and Bob, each with an identical pad of secret keys, a new key on each page in the pad. When Alice sends a message, she uses the top sheet of the pad, using as many key symbols as there are symbols in her message. Once a key sheet is used one time, it is torn off and destroyed.
 - Shannon showed that the one-time pad is perfectly secure; it is a code that cannot be broken by cryptanalysis.

A Binary Version of the One-Time Pad

- ▶ Let’s discuss a binary version of the one-time pad concept. Imagine that Alice’s plaintext is a string of bits. The key is a second, randomly generated string of bits, secret but shared by Alice and Bob. To generate the ciphertext, Alice combines plaintext and key bits using the XOR operation. As you recall, 0 XOR 0 and 1 XOR 1 both yield 0, while 0 XOR 1 and 1 XOR 0 yield 1. Because the key could be anything, any particular ciphertext string is consistent with every possible plaintext message.

Plaintext:	01101000011001010110110001110000
Key:	11100011000010001110100011110100
Ciphertext:	10001011011011011000010010000100

- ▶ To recover the plaintext at his end, Bob simply applies the XOR operation again, using the key string that he also has. That gives him “ciphertext XOR key,” which is “plaintext XOR key XOR key.” Because anything XOR itself is zero, this just yields the plaintext again.

Ciphertext: 10001011011011011011000010010000100
Key: **11100011000010001110100011110100**
Plaintext: 01101000011001010110110001110000

- ▶ If Alice and Bob tried to reuse the key, this binary version of the one-time pad would no longer be perfectly secure. This fact was the basis for one of the most remarkable episodes in the history of cryptanalysis: the Venona project.
 - During the 1930s and 1940s, the Soviet secret intelligence service, the NKVD, communicated with its stations in Soviet embassies around the world using a one-time pad system for ordinary commercial cable messages. However, for reasons that are not entirely clear, identical code key sheets were sometimes used in different pads.
 - Ultimately, the U.S. Army Signal Intelligence Service realized that the same key pages were being reused in different locations and started a project, Venona, to exploit this mistake and decrypt some of the cables. Even though the Soviets were eventually tipped off to the project and switched to new codes, Venona went on for decades.
 - The decrypted Venona messages can be read online and offer a fascinating window into the period. They are also eloquent testimony to a fundamental fact about cryptography: The security of the one-time pad depends on a continual supply of new secret key data; otherwise, it is as vulnerable as any other system.
 - In other words, the problem with the one-time pad is new key distribution. This function cannot be done using the cryptographic system itself, because that uses up as much key data as it would transmit. Some other means, such as trusted couriers, must be used. But as a practical matter, that approach is a point of vulnerability.

Remote Keys and Rolling Codes

- ▶ The one-time pad is reminiscent of the remote keys we now use to open car doors and garage doors. When you press a button on the remote unit, it sends a coded radio signal, a string of a few dozen bits, as the key. The receiver recognizes the key signal and unlocks the door.
- ▶ The problem with this system is that someone nearby could intercept the transmission, record the key signal produced by the remote unit, and re-transmit the same key later to unlock the door. Initially, remote key units used the same key code repeatedly, but modern systems use something called a **rolling code**. Each time you use the unit, it emits a new key code; no key code can open the door more than once.
- ▶ Although this system seems similar to the one-time pad, there's a difference in the way new keys are generated. Both the remote unit and the receiver have, in effect, simple computers that are programmed to produce the sequence of key codes. This is a strict mathematical sequence—not really random at all—based on an internal key-generating code that both units share.
- ▶ The new keys are not really new information, and if someone recorded a few of the successive key signals, it would be possible, at least in theory, to deduce the key-generating code and predict the next key. But that would be a difficult mathematical problem to solve, and the difficulty is what makes the system secure.

Computational Complexity

- ▶ According to Shannon's theory of cryptanalysis, once an eavesdropper intercepts enough message data—a length given by the unicity distance—then there is only one plaintext consistent with the ciphertext data. Once a would-be intruder catches a few key codes from the remote, there is only one secret key-generating code that could produce them; therefore, it is possible to break the code. But Shannon's theory says nothing about how difficult the code-breaking process will

be. Most modern cryptographic systems—the ones used by remote keys, computers, and phones—rely on what computer scientists call *computational complexity*.

- ▶ Any calculation is made up of a series of basic steps, although some calculations require many more steps than others. Some require so many steps that no imaginable computer could finish them in a million years. Such problems are solvable in principle but not in practice. The factoring of large prime numbers represents such a problem.
 - Computers can multiply two large prime numbers very quickly. Even multiplying primes of 70 digits takes only a few hundred nanoseconds. But factoring the 140-digit product of those two primes would take a regular computer longer than the lifetime of the universe.
 - In the past decade, hundreds of computers working together managed to factor a 232-digit number in about 2 years.
- ▶ Factoring prime numbers is what computer scientists call a *trapdoor function*. There is an input—two primes—and an output—their product. Given the input, it is easy to compute the output. Given the output, it is also possible, in principle, to compute the input.
 - In other words, from the point of view of information theory, the input and the output contain exactly the same information.
 - But in the trapdoor function, going backward from output to input is vastly more difficult, requiring many more basic steps and far more time. It might, for all practical purposes, be impossible. Once you pass through the trapdoor in one direction, it is extremely difficult to go back.
 - It is just barely possible that there is a clever, quick method of factoring numbers that no one has yet discovered. Mathematicians have been thinking about factoring for a couple of thousand years, but maybe they've missed something!
 - The fact is: No one has ever proved that a true trapdoor function exists. It may be the most famous unsolved problem in mathematics.

- ▶ In computational complexity theory, NP stands for computational problems that are easy in at least one direction. P stands for computational problems that are easy in both directions. All of the problems of type P are obviously also of type NP. But are there NP problems that are not P? That's what we would need for a trapdoor function.
 - After decades of research into mathematics and computer algorithms, many computational problems seem to be easy in only one direction. That is, they seem to be problems of type NP but not of type P.
 - Although no one has ever quite proven that NP is not the same as P, almost everyone is prepared to assume that they are different. Trapdoor functions do exist, and the factoring problem is one of them. This mathematical hypothesis leads us to an astounding new form of cryptography.

Public-Key Cryptography

- ▶ In the early 1970s, new methods were invented by three British cryptographers: James Ellis, Clifford Cocks, and Malcolm Williamson. Several years later, computer scientists and cryptographers in the United States, including Ralph Merkle, Whitfield Diffie, and Martin Hellman, followed the same path and established the basic theory. Then, Ron Rivest, Adi Shamir, and Leonard Adleman created the most popular and usable version of the system, known as RSA after their initials.
- ▶ What they all discovered came to be called **public-key cryptography**, a system that solves the problem of key distribution. It allows phones, web browsers, and e-mail systems to communicate privately over public networks, and its privacy comes from a trapdoor function.
- ▶ The essential idea of public-key cryptography is to separate the processes of encryption and decryption.
 - The mathematical process of encryption—turning plaintext into ciphertext—uses a number that is the product of two prime numbers. That product is the encryption key.

- However, the process of decryption requires the prime number factors themselves. They make up the decryption key.
- In principle, anyone who has the encryption key can work out the decryption key—you can always factor a number. However, if the number is very large, it might be impossible in practice to do so.
- ▶ Suppose Alice wants to send a secret message to Bob. First, Bob generates two very large prime numbers and multiplies them together. He sends the product, the encryption key, to Alice. He does not have to do this secretly; it's a “public” key.
 - Once she receives the key, Alice uses it to encrypt her message and sends it to Bob. Bob, who has the original two prime factors, can then easily decrypt the message.
 - Eve knows the public key and the ciphertext that Alice sends to Bob. In fact, she has all the information she needs! All she has to do is factor the encryption key to get the decryption key, then decode the message. But if the key is long enough, no computer available to Eve could do the factoring; for this reason, the message remains secret.
 - For this system to work, Bob must generate two very large prime numbers—dozens of digits long—to create the key. Because proving that a particular number is prime is the same thing as factoring it, this step can be tricky. But it's possible to identify primes with a high degree of certainty by applying some mathematical tests.
 - Factoring large numbers is only one possible basis for public key encryption, though it is the one used in RSA and most other systems.
- ▶ Public-key cryptography has applications beyond keeping secrets, such as proving identity at a distance or providing digital signatures, enabling us to conduct business over the Internet. These applications are different than the issues that were important during World War II and the Cold War. The tools we used to conceal information are now being used to establish truth.

TERMS

one-time pad: An unbreakable cryptographic system in which shared key information is used only once and in which the essential problem is the secure distribution of the secret key information.

public-key cryptography: A revolutionary method of cryptography that uses separate encryption and decryption keys. The encryption key can be published, but only the designated receiver has the decryption key. This completely avoids the problem of key distribution. In principle, it is always possible to deduce the decryption key from the published encryption key, but in practice, the necessary computation (factoring large integers into their prime factors) may be too difficult to accomplish.

rolling code: Method used by many remote key systems of generating a new key sequence for each press of the key button. Key sequences are generated by a fixed mathematical algorithm, which could in principle be deduced by monitoring repeated key signals. See also **one-time pad**.

SIGSALY: An encrypted and perfectly secure telephone system used for high-level communications during World War II, in which identical phonograph records of noise were added to and subtracted from a digitally encoded voice signal.

READING

Singh, *The Code Book*, chapters 6–7.

QUESTIONS

- 1 How might a cryptanalyst recognize that a one-time pad key has been used more than once? You may assume that the cryptanalyst is very patient or has the services of a computer.

- 2 A computer system must be able to recognize the passwords of its users. However, if the passwords were stored in some data file in the system, someone who once gained access to the files could obtain everyone's password. Suggest how public-key encryption can reduce this vulnerability.

What Genetic Information Can Do

So far, the science of information has been all about human communication and human technology. But nature has her own messages, her own vast networks of information exchange, her own natural codes for representing information and technologies for processing it. Thus, the concept of information is indispensable for understanding the natural world. And nowhere is this truer than in biology, the science of life.

Early Work in Genetics

- ▶ As we know, living things reproduce, and the offspring resemble the parents. Therefore, there must be some kind of information that is passed from one generation to the next. But what kind of information is this, and how is it represented physically? How is it transmitted? What does it say?
- ▶ We touched on this topic earlier, when we mentioned John von Neumann's idea of a self-reproducing robot. A machine that can manufacture copies of itself must contain a kind of blueprint—information that is itself copied and passed on to the robot offspring. The same must be true of living things.
- ▶ The first person to understand this in a modern way was a monk from what is now the Czech Republic: Gregor Mendel. Mendel did experiments on heredity in pea plants. He identified seven inheritable characteristics for his plants, such as the color and shape of the peas. The first surprise he found was that each characteristic is binary. The peas might be yellow or green but not yellowish-green. Thus, it seems as though each characteristic corresponds to just 1 bit of genetic information.

- Mendel's second discovery, however, was that each characteristic corresponds to 2 bits. Consider pea color. Let 0 represent green and 1 represent yellow. Each pea plant has 2 bits of pea color information, 1 from each parent. But one trait, yellow, is dominant for the actual appearance of the plant. If the bits of the plant are 10, 01, or 11, then the peas come out yellow; if the bits are 00, the peas are green. Those units of inheritable information later came to be called *genes*.



GREGOR MENDEL

- Mendel published his work in 1865, but it was quickly forgotten. When it was rediscovered a few decades later, the science of genetics was born. The location of the genetic information within a cell was soon identified: the chromosomes, tiny structures within cells, barely visible in a microscope.
- Chromosomes occur in pairs, just like Mendel's genes, and their existence suggested an answer to a genetic puzzle: why some pieces of genetic information are transmitted independently and others are linked. Independent traits are carried on different chromosomes, while linked traits are carried on the same chromosome. Geneticists began to map out which traits existed on which chromosomes, but the nature of the gene still remained a mystery.

Schrodinger's Aperiodic Crystal

- ▶ Among those who speculated on the nature of the gene was the Austrian physicist Erwin Schrodinger. He noted that genetic information is discrete rather than continuous; that is, it is digital information, not analog. The first piece of evidence for this idea was Mendel's discrete binary genes in pea plants. The second was a clever argument about the stability of genetic information.
- ▶ At the microscopic level, all matter, including the matter in genes, is subject to continual thermal agitation. Analog genetic information would be subject to continual tiny alterations that would add up over time, but we do not observe that sort of change.
- ▶ Schrodinger noted the experiments on mutations by the American geneticist Hermann Muller. In his work, Muller exposed fruit flies to X-rays. Occasionally, this produced changes in genetic information that were passed on to the flies' descendants. Those changes were discrete, all-or-nothing effects: Either a mutation occurred, or it did not. This led to the conclusion that genetic information is digital.
- ▶ Based on these facts, Schrodinger argued that genetic information had to be stored in molecular structures within the chromosomes. Given the laws of quantum physics, only certain combinations and arrangements of atoms are possible. These structures are ordinarily stable, but individual X-ray photons could make discrete changes in them.
- ▶ Schrodinger described the genetic material as an *aperiodic crystal*. A crystal is a regular, periodic arrangement of atoms. It has a stable structure, like the gene, but because the pattern of a crystal endlessly repeats itself, it contains almost no information. Therefore, the molecular units in the genetic material must have a more complex nonrepeating arrangement, which serves as a code for genetic information.
- ▶ Schrodinger's aperiodic crystal amounts to a new state of matter: neither the perfectly regular (but information-less) structure of a crystalline solid, nor the unstable, disorderly chaos of a liquid or a gas.

Crick and Watson's Discovery

- ▶ Schrodinger's work attracted the attention of Francis Crick and James Watson. Working together at Cambridge University in 1953, they unraveled the aperiodic structure of the molecule within the chromosome that carries the genetic information: DNA.
- ▶ Crick and Watson found that two “backbones” of phosphate and sugar form the sides of the DNA molecule. In three dimensions, the structure is twisted in a double-helix form, but we can think of the backbones as the two sides of a simple ladder.
- ▶ Between the backbones—the rungs of the ladder—are pairs of flat molecular units called *bases*. There are four bases: adenine, guanine, thymine, and cytosine, which are represented by the letters A, G, T, and C.
- ▶ These fit together in complementary pairs—A fits with T and G fits with C—and the pairs can be oriented within the molecule either way. Each rung of the DNA ladder is either A-T, T-A, G-C, or C-G.
- ▶ The structure, then, is just as Schrodinger guessed: a regular, stable molecular structure with units arranged in a nonrepeating pattern. That is the genetic information for the organism, written in a kind of molecular code.
- ▶ Crick and Watson were quick to note that the base-pair system makes copying straightforward. The original DNA molecule first splits apart along the “join” between the bases in each rung. New units assemble along the split halves, but only certain bases can fit together: A with T or C with G. Thus, the original sequence of base pairs—the genetic information—is reproduced in the two resulting DNA molecules.

The Nature of the Genetic Code

- ▶ A living cell contains only a tiny amount of DNA, but that still may represent billions of bits of information. The DNA information in a cell is a blueprint of its structure and function. What kind of code is used for that? One of the first people to glimpse the answer to this question was the physicist George Gamow.
- ▶ In his work, Gamow focused on proteins, the basic molecular “machines” of a cell. A protein is a long polymer or chain composed of a few hundred basic units called *amino acids*. There are many different types of amino acids, 20 of which appear in proteins. The protein molecule is not physically stretched out in a line but folded up in a compact, complicated, three-dimensional shape. The details of that shape determine what the protein does.
- ▶ Gamow believed that the sequence of base pairs in the DNA molecule must somehow encode the sequence of amino acids in the protein.
 - The “words” of the protein language are the amino acids. Given that there are 20 possible words, each amino acid represents $\log_2(20) = 4.3$ bits of information. But the DNA language has only four letters—the base pairs—for 2 bits per letter.
 - Thus, Gamow conjectured that in the genetic code, each amino acid was represented by a codeword, called a **codon**, consisting of three DNA base pairs. Later experiments by Francis Crick and others showed that he was exactly right.
 - The DNA “alphabet” has four letters. The codon “words” are three letters long, representing the 20 amino acids. These form long “sentences”—proteins made up of hundreds of amino acids. One of those sentences is one of Mendel’s genes, because the proteins control the way the organism appears and functions.

- ▶ DNA itself does not play a direct role in forming proteins. DNA data storage and protein construction may occur in entirely separate parts of a cell. The molecule that carries the information from one to the other is called **RNA**—a kind of one-sided cousin of DNA.
- ▶ RNA has is a single phosphate-sugar backbone, and individual bases are attached to it on one side. These are the same bases as in DNA, with one exception: Thymine (T) in DNA is replaced by uracil (U) in RNA. Thus, the RNA alphabet is A, U, G, C.
- ▶ The base sequence in a strand of RNA represents genetic information, just as in DNA. RNA acts as an intermediary between the data storage and the protein making. In the process of *transcription*, a DNA molecule partly unzips and an RNA molecule forms from complementary base pairs. Then, this messenger RNA (mRNA) moves to a ribosome, where proteins are made. That process is called *translation*.
- ▶ Because the two ends of the RNA molecule are slightly different chemically, the ribosome knows which end to start with. Each group of three bases is a codon, but the first codons on the RNA strand might not represent anything. They are in the *untranslated region* of the RNA molecule. Protein building does not start until a special triplet called the *start codon* is reached. There is some variation, but usually, this is the codon AUG, which stands for the amino acid methionine.
- ▶ Each triplet of bases—each codon—is translated as the next amino acid in the protein. Of course, because there are no spaces or commas in the molecule, a sequence of bases could be divided up in three different ways.
 - Imagine a stretch of RNA with a repeating pattern of C, A and U. This could be read as:

... AUC AUC AUC AUC AUC AUC AUC ...

... A UCA UCA UCA UCA UCA UCA UC ...

... AU CAU CAU CAU CAU CAU CAU C ...

- The three ways of dividing the sequence into codons are called different *frames*, and only one of them is correct. The right one is determined by the start codon. A mistake in the RNA that inserts or deletes a single base will produce a *frame error*, but that usually does not lead to a sensible protein.
- The translation continues until one of three possible “stop” codons is reached: UAA, UAG, or UGA.
- ▶ If we count the codewords, we find that four bases, taken in groups of three, give us $4 \times 4 \times 4 = 64$ possible codons. Three of these mean “stop.” Every one of the remaining 61 codons, including the “start” codon AUG, stands for some amino acid. There are no “nonsense words” in the genetic code, but there are only 20 different amino acids used in natural proteins. That means that most of them can be represented by two, three, or more different codons. The genetic code, in other words, is somewhat redundant.

Advances in Understanding DNA

- ▶ The details of the genetic code were worked out in the early 1960s, but since then, some dramatic technological advances have been made. We have learned how to read and create very long DNA sequences. We can record the complete genome—the entire collection of genetic information—of an organism.
- ▶ Our own genome contains 6 billion base pairs, or about 12 billion bits. Only about 2% of that, however, actually describes proteins via the genetic code. The rest is called *noncoding DNA*.
 - For instance, the genes coding proteins are very far apart, separated by long stretches of noncoding DNA. That means that they are always translated separately, and that means that a frame error—a base lost or inserted—affects only one of the proteins, not both.

- Not all organisms have as much noncoding DNA as we do. In bacteria, for example, about 80% of the DNA codes for proteins.
- ▶ Chemically, there are many more than 20 possible amino acids. Thus, there is a universe of conceivable proteins, some of them perhaps useful, that cannot be made by ordinary living things on Earth. We don't know how to make them because the genetic code has no room for them; we've used up all the codewords. Can we change that? Can we find a way to alter cells so that they can use “unnatural” amino acids in their proteins?
 - The translation of a codon in mRNA into an amino acid is done by a short piece of RNA called *transfer RNA* (tRNA). A tRNA molecule is bent in a kind of hairpin shape, with side loops. Complementary bases attach to each other along the parallel sides.
 - At one end, where the hairpin bends around, three bases stick out. These three bases can attach themselves to a complementary set of bases on the mRNA chain: U with A, G with C, and so on. And each tRNA molecule is bound at the other end to a particular amino acid.
 - Thus, as a tRNA molecule attaches to the codon on the mRNA strand, it can add the next amino acid to the protein. That's what happens in the ribosome protein factory.
 - To incorporate new amino acids, creating new kinds of protein that no existing life could ever make, we'd need some new tRNA modules with the new amino acids attached. We would also need to expand our genetic code.
 - Molecular biologists have approached this problem in several ways, including repurposing an existing codon or using the “stop” codon UAG. In fact, a great many proteins with “unnatural” amino acids have been made in the lab using exactly these kinds of modified cells.

- ▶ Another way to expand our genetic code would be to use longer codewords. We could imagine codons with four or even five bases. The tRNA molecules that recognize them will have somewhat wider ends with more matching bases exposed.
- ▶ Finally, and most radically, we can imagine nucleic acids with more different bases. Our genetic code could then employ a larger alphabet. In addition to A, U, G, and C, we might have X and Y. The number of possible codons is now $6 \times 6 \times 6 = 216$ codewords, which gives plenty of room for new amino acids in the new code. This has actually been done for the DNA in some bacteria.
- ▶ These experiments on the information machinery of life can seem a little weird and uncanny. They are like visits to a different planet, where life uses similar DNA and RNA but a completely different code for its genetic information. Codes are, after all, arbitrary associations between symbol and message. Why, then, does our own genetic code—which we share, in almost every detail, with every organism on Earth—have the structure that it does? As it turns out, the code we use has some especially nice features.
 - Consider the redundancy of the genetic code, the fact that most amino acids are represented by many different codons. This makes the code more tolerant of errors.
 - For example, the codon ACC translates as the amino acid threonine. But suppose something happens to the RNA and the final letter is changed to something else. The codon becomes ACA, ACG, or ACU. Any of these still translates as threonine. The protein is unaffected.
 - Changing one of the other letters will change threonine to some other amino acid. That is an error, because it affects the resulting protein, but not all of those changes would necessarily be disastrous. Changing the first letter from A to U (yielding UCC, UCA, UCG or UCU) changes the amino acid from threonine to serine, but those two are chemically quite similar. A protein in which one is substituted for the other might have a similar shape and function. It is a mistake, in other words, but perhaps not a serious one.

- ▶ It's wrong to imagine the molecular machinery of life as a bunch of well-oiled mechanisms in which the gears all mesh together perfectly and everything proceeds step by step. In reality, all the molecules jitter around in ceaseless thermal motion. That agitation can drive processes forward and help different molecules "find" each other in the soup. But the progress is unsteady, sometimes going two steps forward and one step back or producing errors. The genetic code must be able to work in that sort of environment.
- ▶ We've used computer models to see how our own genetic code stacks up against other conceivable genetic codes, generated at random. As it turns out, our code tolerates errors better than the vast majority of the others. Our basic genetic information system is well-adapted to the noisy molecular reality.
- ▶ Of course, errors are a kind of information too. So far, we've discussed errors in RNA and protein-making, but there can also be errors in DNA copying—the process by which genetic information is passed from one generation to the next. As long as the errors are not fatal, they will, in turn, be transmitted to later generations.
 - By looking at the patterns of DNA variation between individuals, we can draw conclusions about their common family tree. Apply this to whole populations or species, and we get a bigger picture of the pattern of descent—what geneticists call a **phylogenetic tree**.
 - The Italian geneticist Luigi Luca Cavalli-Sforza and his colleagues have applied these ideas to our own species, reconstructing the phylogenetic tree of human populations. Their work gives us a glimpse of the history of humanity, the great human diasporas by which our species spread over the Earth.
 - In the last two decades, biologists have begun to use DNA to find the broad outlines of the phylogenetic tree of all living things—that is, the tree that includes everything from bacteria to begonias to buffalo. At the root of that tree is what's known as the *last universal common ancestor* (LUCA), which probably lived 3.5 or 4 billion years ago.

TERMS

codon: A group of three successive bases in DNA, forming a codeword that represents a single amino acid in a protein. With four base pairs (AT, TA, CG, and GC), there are 64 possible codons, which is more than enough to represent the 20 possible amino acids. (There are also “start” and “stop” codons to designate the ends of the protein sequences.)

phylogenetic tree: A tree structure representing how information flows from an original to generations of offspring, which can often be reconstructed by examining the copying errors that occur as the information is passed along.

RNA: Ribonucleic acid, a sort of one-sided cousin to DNA. *Messenger RNA*, or mRNA, carries information from DNA to the ribosomes, where it is used as template for protein construction. *Transfer RNA*, or tRNA, is a short, looped form of the molecule that is attached to amino acids and helps to recognize the codon sequences in mRNA. RNA molecules can also act as biochemical catalysts, called *ribozymes*.

READINGS

Cavalli-Sforza and Cavalli-Sforza, *The Great Human Diasporas*.

Gleick, *The Information*.

Schrödinger, *What Is Life?*

Siegfried, *The Bit and the Pendulum*, chapter 5.

QUESTIONS

- 1** In DNA, the distance along the ladder between adjacent base pairs is 0.33 nanometers ($1 \text{ nm} = 10^{-9} \text{ m}$). What is the length of the entire human genome? Which is longer, you or your DNA?

- 2** On an alien planet, we find that life uses DNA to store genetic information, but only one pair of bases (A and T) is used. In the genetic code, there are four bases per codon. What is the number of possible amino acids available to life on this planet? Comment on the fault tolerance of the alien genetic code.

Life's Origins and DNA Computing

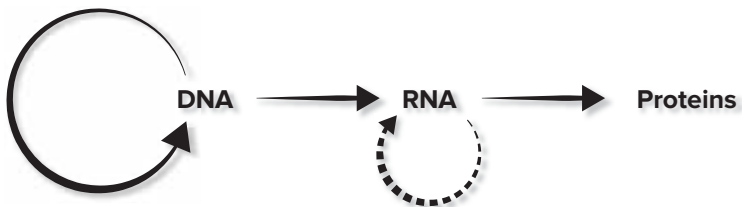
In the past, people thought that living organisms could emerge from nonliving matter. This idea was called *spontaneous generation* or *abiogenesis*. Frogs and worms might spring from mud; mice could arise from discarded food and rags. But the 17th-century Italian naturalist Francesco Redi argued the opposite. Every living thing, he said, is always the offspring of another living thing. As he put it, “*Omne vivum ex vivo*”—“All life comes from life.” And by the end of the 19th century, such biologists as Louis Pasteur had established the truth of Redi’s principle, even for the tiniest microorganisms. Yet that leaves us with a question: What is the origin of life? Where did the first genetic information come from?

The Central Dogma of Molecular Biology

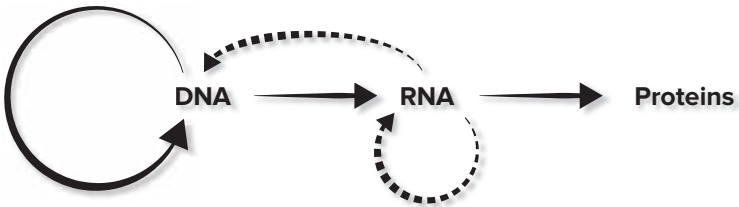
- ▶ Genetic information is stored in the sequence of bases in the double-stranded structure of the DNA molecule. That two-sided, complementary structure is ideal for copying genetic information. DNA also serves as a template to create RNA, the simpler one-sided cousin of DNA, which in turn forms the blueprint for protein molecules. The following diagram outlines this process:



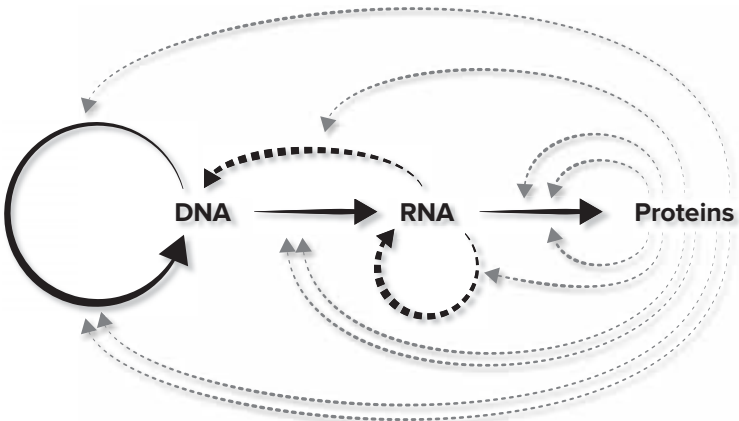
- ▶ The arrows in the diagram represent the flow of information, how the genetic message is transferred from one kind of molecule to another. The information in DNA is copied into other DNA, and it is transcribed into RNA, then translated into proteins.
- ▶ Francis Crick called this the central dogma of molecular biology: Information flows from DNA and RNA to proteins, not vice versa. Base sequences are translated into chains of amino acids, according to the genetic code. Chains of amino acids are never translated back into base sequences.
- ▶ The central dogma is true, but it's more complicated than it seems. Consider, for example, viruses.
 - A virus injects its own genetic information into a cell. There, it is copied and transcribed, just like the cell's own genes, and the cell ends up making more viruses.
 - Some viruses use DNA as their genetic material, which means that the information flows exactly as we have shown: DNA to RNA to proteins.
 - But other viruses, called *RNA viruses*, dispense with DNA altogether. Their genetic material is RNA. That RNA is replicated, along with the virus's proteins, inside the cell. This process changes our diagram:



- ▶ The retroviruses are even stranger. They also carry their genetic information in the form of RNA. But inside the cell, that RNA undergoes something called *reverse transcription*, in which the RNA is used as a template to produce DNA. This DNA is actually incorporated into the host cell genome, after which, the virus can be impossible to eliminate. HIV, the human immunodeficiency virus, is an example of a retrovirus. Again, this changes our diagram, adding another arrow from RNA back to DNA.



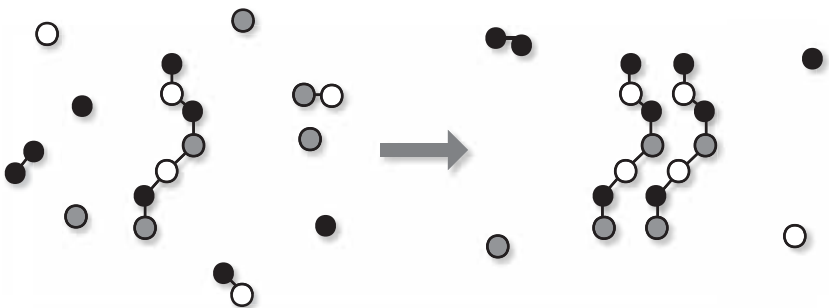
- ▶ But even this picture leaves out something important. Proteins are the “worker bees” in the biochemistry of a cell. They form a good deal of the structure of the cell, and they act as enzymes.
 - An enzyme is a biological molecule that works as a catalyst, assisting a chemical reaction; they break molecules apart or join molecules together. The biochemistry in a cell simply cannot take place without a number of different enzymes. And almost every enzyme is a protein.



- Each arrow in our information diagram—copying, transcription, translation—is a biochemical process that involves taking molecules apart and putting them together. Each one is attended by a swarm of specialized enzymes—proteins—that make it all happen. Thus, in reality, proteins are present everywhere in the diagram.
- ▶ Here is the larger point: The basic information system of life is complex and interdependent. Proteins can't be made without nucleic acid blueprints (DNA and RNA), and DNA and RNA can't be made without protein enzymes. How does such an interdependent system get started in the first place?

Molecular Beginnings

- ▶ Let's begin with a molecule. It's a long chain of small units—a kind of polymer. The units are of two or more different kinds. The particular sequence of units in the molecule constitutes information. This is a one-dimensional version of Schrodinger's aperiodic crystal, like the sequence of bases in RNA or the sequence of amino acids in a protein.
- ▶ The environment of the molecule contains many of the basic units and even some shorter segments floating around. Interestingly, chemical interactions among the fragments tend to cause units to assemble in the same sequence alongside the original.



- ▶ What we have described is an information-bearing molecule that, in the right environment, can reproduce itself. Offspring molecules get their information from their parents. Molecules with different sequences can compete for resources. We've stripped away all the complications found in present-day organisms, and we're left with a naked genome, living in an environment in which spontaneous copying is possible. Such a molecule might be the beginning of the whole genetic system of life.

Eigen's Error Catastrophe

- ▶ This is just the sort of system that the German biophysicist and chemist Manfred Eigen was thinking about in the 1970s. Shannon had pointed out that any process by which information is transmitted may be subject to noise and result in errors. Eigen realized that errors are important.
 - A self-reproducing molecule makes a copy of itself. Then, those two molecules make copies of themselves, and so on. If the copying process introduces errors, after a while, the vast majority of molecules in the population will have many errors. There will be many different versions of the genetic information, and few, if any, of them will look much like the original.
 - This is not a theoretical problem. Our bodies contain tens of trillions of individual cells, but every one of them is descended from a single cell—a fertilized human egg. The DNA in each of our cells has been copied many, many times since fertilization. Of course, because the cells still have to function, it is important that their genetic information has been copied correctly.
- ▶ The DNA copying mechanism in human cells is a miracle of reliability, with a good deal of error correction. The enzymes that assemble new DNA molecules proofread as they go, actually reversing and replacing bases that have been added incorrectly. Other proteins can later find and repair most of the remaining mismatches between bases. The likelihood that a given base will be copied incorrectly in DNA replication is about one in a billion.

- ▶ But in the early days of self-reproducing molecules, there was no fancy enzyme machinery to speed the copying process along and correct mistakes, and the error rate would have been fairly high. Thus, our self-reproducing molecule would produce defective copies, and their copies would be even more defective. Soon, all of the original information would be wiped out by random noise, leaving us with a population of random molecular sequences, randomly changing over time—interesting chemistry but not life.
- ▶ Eigen realized that there could be a kind of error-correction mechanism, even in the chancy world of reproducing molecules. That mechanism is natural selection. Some molecular sequences might be easier or faster to copy; they might be sturdier and last longer. Those sequences would have an advantage over their imperfect copies, and perhaps that advantage would be enough to preserve their information over time.
- ▶ Let's suppose that a particular sequence is S times more likely to successfully survive and copy itself than other sequences. The overall length of the molecule, the number of information units it has, is L . When a unit is copied, there is a probability e of an error. Here's what Eigen showed: The genetic information of a sequence is stable in the long run provided that $Le < \log S$. If Le is greater than this, then over time, errors will destroy the information, and it will become extinct. That is called the ***error catastrophe***.
- ▶ If $S = 1$, the right sequence is no better than any other sequence; $\log S = 0$, and the error catastrophe is inevitable. If $S > 1$, then some combination of length L and error rate e is acceptable. But even if S is huge—if the right sequence is thousands of times better than its fellows— $\log S$ is still not very large—less than 10. A more realistic guess for $\log S$ would be around 1. That is some advantage but not a huge one. We thus expect that Le cannot be much greater than 1.
- ▶ That should be true not only billions of years ago but also today. We said that the error rate for the human genome is something like one-billionth: $e = 10^{-9}$. The size of the human genome is about 6 billion

base pairs: $L = 6 \times 10^9$. The product Le is 6, which is not that much larger than 1. (And remember, most human DNA is noncoding. The total length of the genes that code for proteins is 50 times smaller.)

- ▶ The process by which virus DNA or RNA is copied is much more prone to error. The error rate is about 1 in 10,000. But viruses don't need much genetic information. The genome of a typical virus is only about 10,000 base pairs long. The product Le is about 1—just small enough to pass.
- ▶ Eigen guessed that in the earliest days of life, the error rate might be around 1%. That would mean that any successful self-reproducing molecule could not be much longer than 100 units. Otherwise, it would be overtaken by the error catastrophe.

Spiegelman's Monster

- ▶ This number, 100 units, is interesting. Not long before Eigen's work, the American molecular biologist Sol Spiegelman did an interesting experiment on self-reproducing molecules. He started with a very simple RNA virus called Q-beta. The RNA genome of this virus is only 4200 bases long, containing descriptions of just four proteins.
- ▶ Spiegelman extracted the RNA from Q-beta and placed it in a test tube containing a solution of the raw materials to make RNA, plus a supply of the RNA replicase enzyme—the same enzyme the virus uses to reproduce itself inside a cell.
- ▶ The RNA in the test tube began to make copies of itself. Spiegelman repeatedly transferred tiny samples from one test tube to another. The time between transfers was only a few minutes. In other words, Spiegelman was arranging conditions to favor RNA molecules that reproduced quickly.
- ▶ Interestingly, the RNA molecules started getting shorter. The RNA in Spiegelman's test tubes no longer had to travel from bacterium to bacterium. It no longer needed to manufacture its own replicase

enzyme. It was in molecular paradise! Thus, the molecules could afford to lose sections of themselves if they could thereby become faster and easier to copy.

- ▶ After 74 stages, Spiegelman's RNA molecules were only 220 bases long, but the information in those RNA sequences was fairly stable. The molecules were fiercely efficient replicators, just large enough for the replicase enzyme to work. This shortened, no-frills RNA molecule came to be called *Spiegelman's monster*.
- ▶ Spiegelman's experiment was very artificial, but it tended to confirm Eigen's rough estimate—and that is a problem. If the earliest precursors of life were limited to 100 or 200 units, how could more complex life develop? The error rate must somehow be reduced. Modern organisms keep the error rate low by using protein enzymes, but 100 information units is not enough to code for any useful protein. The error catastrophe seems like an impassable barrier.

The RNA World

- ▶ In 1982, Thomas Cech of the University of Colorado made a game-changing discovery. Small RNA molecules can be folded up in specific shapes. In fact, RNA molecules can act as chemical catalysts—enzymes. They're called *ribozymes*, and they are far more than mere chemical curiosities. The basic protein-building factories inside cells, the ribosomes, are made up of both proteins and ribozymes. RNA catalysts do exist and play a key role in the central biochemical machinery of all life. Thus, perhaps the original molecular information diagram looked like the one at right, showing the genetic information in RNA copied with the help of RNA catalysts.
- ▶ This notion was christened the *RNA world* by another Nobel laureate, the chemist and physicist Walter Gilbert. In the RNA world, the first life—the carrier of the earliest genetic information—might have been a collection of relatively short sequences of RNA.

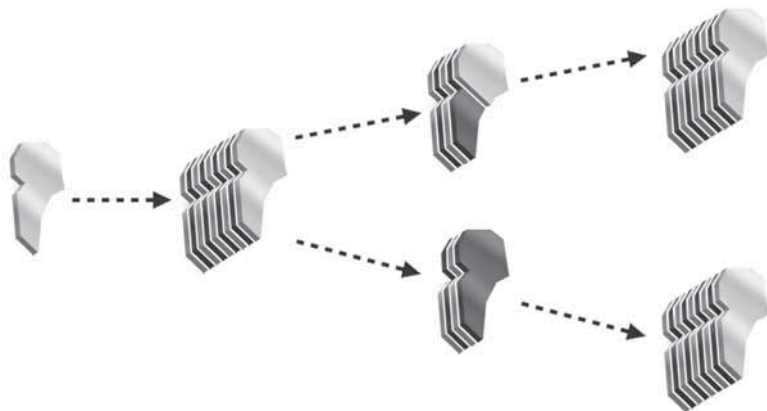


- ▶ Each sequence was short enough to avoid the error catastrophe. Some of the sequences acted as primitive ribozymes, catalyzing the assembly of new RNA strands of different kinds. The whole thing was a mutually supporting system of molecules—what Eigen called a *hypercycle*. Very likely, the molecules were weakly attached to mineral surfaces or gathered inside the cavities in porous rocks. Those surfaces may have contributed some catalytic action of their own.
- ▶ The RNA system changed as new sequences appeared and outcompeted the others. Other complications arose later, including proteins, which can act as much more efficient enzymes, and DNA, which is a more stable way to store genetic information—the information is on the inside of the molecule instead of the outside. Little by little, the complex interdependent machinery of life was built up.
- ▶ The RNA world is an attractive hypothesis, and there have been some recent discoveries that support it.
 - Many scientists argue that Eigen's estimate of the early error rate—1%—is probably too large. That would imply that longer RNA sequences could escape the error catastrophe.
 - At Cambridge University, Philipp Holliger and his colleagues have been looking for new simple RNA catalysts. They do this by a kind of artificial molecular evolution, like Spiegelman's experiment, but they are looking for an RNA molecule that can help other RNA molecules to copy themselves—a kind of *replicase ribozyme*. Interestingly, many of these ribozymes work best in very cold water, attached to ice crystals.
 - For a long time, the ribozymes Holliger found could only catalyze the copying of RNA sequences much shorter than themselves. But very recently, the researchers have discovered a new mutant molecule, which they call *tC9Y*. This molecule is 202 bases long, but it can assist the copying of some RNA strands up to 206 bases long.
 - The dream of the RNA world, a world of nucleic acids that catalyze their own reproduction, seems more plausible than ever.

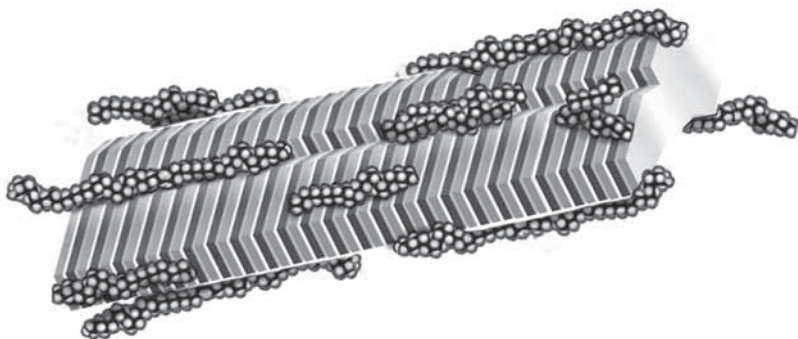
Self-Reproducing Clay Crystals

- ▶ Not everyone, however, thinks that it all started with RNA. Graham Cairns-Smith, a molecular biologist at the University of Glasgow, looks at the problem this way: The biochemical system of life is like an arch. Every segment of an arch rests on the others. If you take any one out, it all collapses. Given that, how could the arch have been built piece by piece?
- ▶ The answer is that there was a scaffold, on which the pieces of the arch were assembled. Once the arch was complete, the scaffold was removed, and we no longer see it. We just see the complete, self-supporting arch.
- ▶ Cairns-Smith believes that the earliest form of life was very different from us. It stored information, grew, and reproduced in a completely different way. But it served as a scaffold, on which the other pieces—DNA, RNA, and proteins—were added. The new biochemical machinery turned out to be much more efficient. It developed a complex system of its own and eventually became self-supporting. Thus, it took over, and the old form of life crumbled away.
- ▶ Cairns-Smith believes that this lost scaffold for life might have been crystals of clay. The mineral kaolin, which is a silicate of aluminum, is extremely common. It forms readily in water solutions. And its crystal structure can store and copy information.
 - An ideal crystal is a perfect, repeating arrangement of atoms; it contains no information. But a real crystal is usually more complicated. In each small piece, the structure is perfect, but the atoms are stacked differently in different places. There are boundaries between these domains.
 - Imagine a flat piece of kaolin crystal, like a plate. Its structure includes several crystal domains, fitted together. The pattern of these domains constitutes information.

- Now, the mineral grows in water solution, getting thicker. Each new layer replicates the pattern and shape of the layer on which it grows. The crystal is like a stack of identical plates, each with the same pattern. The information is copied. Eventually, the crystal grows long, and something breaks it, but now, each piece can serve as a template for new growth.



- The outer edge of the crystal has a shape that is determined by the pattern of crystal structure. Thus, the growing crystal has grooves and ridges, to which organic molecules can attach. These may form long chains as the crystal grows. In fact, the crystal and the molecules may form a mutually beneficial system, each supporting the growth of the other. But the information is stored and copied in the pattern of the kaolin crystals.



- ▶ There are many factors that make Cairns-Smith's theory plausible. For example, clay crystals are exceedingly common and have a wonderful variety of forms. They can also be excellent catalysts.
- ▶ If Cairns-Smith is right, then the original genetic information on Earth was contained in the patterns of self-reproducing clay crystals. Our kind of information—the information stored in DNA and RNA—arose later and eventually took over. The first framework was lost and, with it, the old crystalline genetic information.
- ▶ This means that our most remote ancestors were, in an information sense, not actually related to us. Yet as Cairns-Smith points out, a rope might be hundreds of feet long, even if no single fiber of that rope reaches from end to end. We can think of the rope as the history of life and the threads as threads of information. But where, exactly, does the genetic information come from?

The Process of Evolution

- ▶ A few years after Sol Spiegelman created his rogue RNA monster, Manfred Eigen repeated the experiment. He first showed that a single strand of viral RNA was enough to get things going. Then he tried starting with no RNA at all. He set up his test tubes with RNA components and replicase enzyme and waited.
- ▶ Sometimes, he found, a short strand of RNA would form spontaneously and start replicating. After a while, it might get longer. Under the right circumstances, we can end up with something very much like Spiegelman's monster. Where does that information come from? The answer is: the environment.
- ▶ When RNA molecules are copied, errors are sometimes made. Genetic information is lost, and random bits are introduced. Natural selection can help fix the mistakes and restore the lost information. This kind of error correction process does not happen at the individual level.

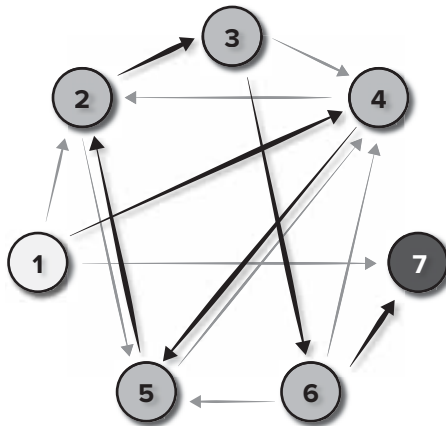
Each molecule is what it is; it's either a faithful copy or not. Natural selection affects the whole population of replicating molecules so that they may collectively avoid the error catastrophe and retain the genetic information.

- ▶ But notice: This process only happens if there is some correspondence between the molecular sequence and the environment, that is, if the original sequence is favored in that environment. Thus, the sequences that survive and beat the error catastrophe actually form a kind of record of the environment in which they succeeded.
- ▶ We have called the process *error correction*, because most of the time, it is conservative, weeding out harmful mistakes in the population. But occasionally, one of those mistakes by chance produces a genome that is a better match to the local conditions—a closer representation of environmental information. Then, the error correction process actually favors the innovation. That's how evolution works; it is also the principle behind one of the strangest computers ever built.

Adleman's DNA Computer

- ▶ In the early 1990s, the computer scientist Leonard Adleman became interested in DNA. He began to think about the complex system of enzymes that molecular biologists use to manipulate the information in the DNA molecule, and he decided to make a DNA computer.
- ▶ The computer was designed to solve something called a *directed Hamiltonian path problem*. Computationally, this is quite a difficult problem, though Adleman chose a very small version of it.
- ▶ Imagine seven cities, numbered 1 through 7, connected with a network of roads. Some of the connections are one way; some go both ways. The problem is to find a road that starts at 1 and ends at 7, having traveled through each city exactly once. That is the Hamiltonian path.

- In the network Adleman considered, there is only one Hamiltonian path: 1 to 4 to 5 to 2 to 3 to 6 to 7.



- Adleman represented the cities by short strands of one-sided DNA, each of them containing coded names. Other one-sided strands represented the roads, with complementary DNA sequences of the cities at the endpoints. He mixed them together in a solution with an enzyme, and in less than a second, the short strands linked together. In fact, they linked together in all sorts of ways, forming billions of strands, each one representing a possible path among the cities.
- Then, Adleman performed a series of steps that ensured that the only DNA strands that would easily make copies of themselves were those with three characteristics: (1) They started at 1 and ended at 7; (2) they included exactly seven cities; and (3) they included every city at least once. The resulting sample of DNA consisted of molecules that encoded the Hamiltonian path. The molecules had found the solution to the problem: 1 to 4 to 5 to 2 to 3 to 6 to 7.

- ▶ Adleman's DNA computer was the first, but it has not been the last. New techniques and experiments have led to increasingly sophisticated DNA computations. Despite the jittery molecular environment and the almost complete lack of control, the molecules evolve their way toward the desired result. There is even a DNA computer that can play tic-tac-toe.
- ▶ How did Adleman's computer find the solution to the directed Hamiltonian path problem? Where did that information come from? It came from the environment of the DNA molecules in his experiment. Those few molecules that happened to encode a correct solution were strongly favored to copy themselves. The resulting molecular population reflected that natural selection.
- ▶ In the same way, the genetic information in living cells is the result of eons of "natural computation." Their DNA contains a deep record of the whole history of life, of the chemical facts and environmental forces that have shaped life at every stage. Some of that record is extremely ancient, billions of years old. With the help of the science of information, we are slowly beginning to read it.

TERM

error catastrophe: Condition for the preservation of the genetic information in a self-replicating molecule, first formulated by Manfred Eigen. If the genome length multiplied by the error rate is much larger than 1, the genetic information stored in the molecule will eventually be lost in the population. This means that for a given error rate, there is a maximum stable length for a genome.

READINGS

Avery, *Information Theory and Evolution*, chapter 3.

Cairns-Smith, *Seven Clues to the Origin of Life*.

Davies, *The Fifth Miracle*, especially chapters 1–5.

QUESTIONS

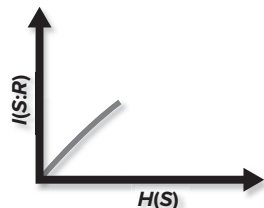
- 1 Discuss the distinction between a true error-correcting code and the error correction in self-replicating molecules provided by natural selection.
- 2 Why is an RNA world hypothesis for the origin of life more plausible than a DNA world or a protein world?
- 3 In the last couple of decades, scientists have discovered prion diseases. These are produced by misshapen proteins called *prions* that act to catalyze the misshaping of other proteins in an organism. The disease can spread, for instance, through eating the meat of an infected animal. Is a prion disease “alive”?

Computers and brains are alike in many ways. They both have inputs and outputs, and they store and process information. However, the basic components of the brain, the neurons, are much slower and more imprecise than the electronic components of a computer. On the other hand, the brain has billions of neurons, all working at once. Also, the brain consumes much less energy than a computer. The differences between the two are significant, yet thinking of the brain as a computer suggests some interesting questions: How many bits of information do our senses provide us? What is the neural code for the electrochemical signals among the neurons? How is memory stored, and what is the memory capacity of the brain? In this lecture, we'll try to map out some connections to learn what the principles of information can tell us about the workings of the brain.

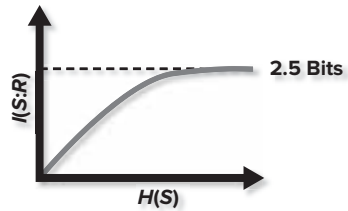
Information Capacity in Humans

- ▶ We can approach the topic of information science and the workings of the brain in two ways: from the top down and from the bottom up.
 - The top-down approach is to begin with the behavior of a subject, an animal or a human, then do experiments and see how the subject responds. The results are clues—indirect and often ambiguous—about what happens to information in the subject's brain.
 - The bottom-up approach starts with a single **neuron** and asks: How does it work? What signals does it transmit? How does it interact with other neurons? Perhaps we monitor the activity of an individual neuron inside a live subject, testing to see how changes in the outside stimulus change the signals.
 - The top-down and bottom-up approaches are complementary. They teach us different things.

- ▶ One of the earliest top-down studies of information in the brain was a classic 1956 paper by the Harvard psychologist George Miller entitled “The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information.”
 - Miller describes experiments by many scientists in which human subjects were taught a collection of different possible stimuli. For example, there might be some tones with different pitches. Then, one of those tones, chosen at random, was briefly played for the subject, and he or she would try to identify which one it was.
 - If the number of possible is tones was low—three or four—the subjects got the right answer all the time, but with more tones, they made mistakes.
- ▶ Things become interesting if we treat this as a problem of information theory. The test subjects are a noisy communication channel. The tone that is played (the stimulus S) is the transmitted message. The subject’s judgment (the response R) is the received message. Mistakes are errors in the channel from S to R .
- ▶ As we know, Shannon’s theory tells us how to measure information. The input information is the entropy $H(S)$. That’s $\log_2(N_S)$, where N_S is the number of possible stimuli (the tones). The amount of information conveyed by the channel is the mutual information $I(S;R)$. That’s $H(S)$ minus the conditional entropy of S given R : $I(S;R) = H(S) - H(S|R)$. This is how much information gets through.
- ▶ We try the experiment many times with different numbers of tones, estimating $I(S;R)$ from the results. Then, we summarize the results by plotting $I(S;R)$ versus $H(S)$, that is, the amount of information that actually gets through versus the total amount of information in the tone.
 - For small values of N_S , the subjects never make mistakes; thus, $I(S;R) = H(S)$.



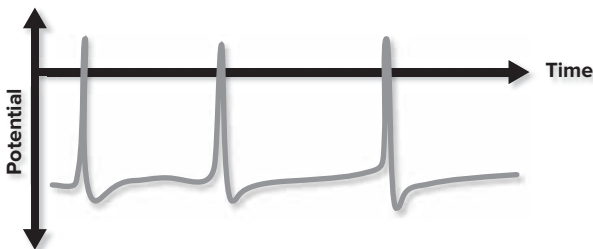
- As $H(S)$ increases, errors creep in; thus, $I(S:R) < H(S)$. The curve bends over and runs horizontally. It maxes out at around 2.5 bits. After that, using more different tones just leads to more errors.



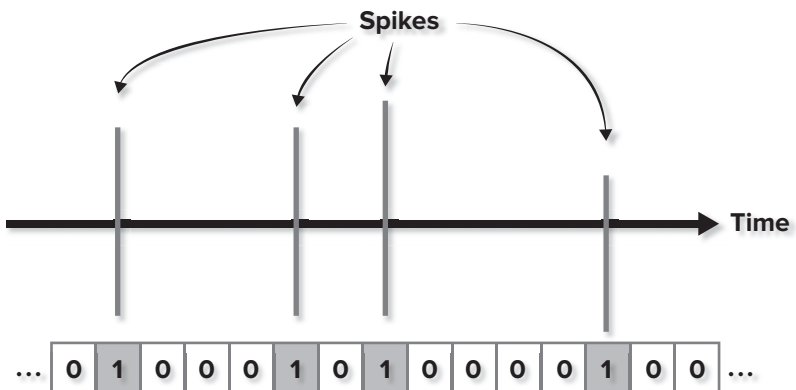
- That's exactly what we would expect if the information capacity of the channel—the subjects' ability to distinguish tones—were about 2.5 bits. That amount, 2.5 bits, is about $\log_2(6)$. Thus, we can reliably distinguish about six tones. More than that, and we start making mistakes.
- Only six tones seems shockingly low. Surely, our ears provide us with more information than that, but something about the way we process the information in this simple task limits our capacity.
- ▶ This experiment can also be done with other types of stimuli. For example, the tones could differ in loudness rather than pitch. Exactly the same kind of curve emerges, except that the plateau—the capacity—is about 2.3 bits, or around $\log_2(5)$. We can reliably distinguish about five levels of loudness.
- ▶ Such experiments involve a one-dimensional stimulus—a stimulus that varies only in pitch or only in loudness. In each of these one-dimensional experiments, we can distinguish between 5 and 9 (7 ± 2) distinct stimuli. That's our capacity, and in general, the same story applies to all kinds of sound, vision, taste, and touch experiments.
- ▶ The simple information capacity picture that George Miller painted has become far more complicated over time, with more experiments of different kinds. Yet it does appear that our brains are well-adapted to make rapid judgments involving only a few bits of perceptual information.

How the Brain Processes Information

- ▶ Neurons are the basic units for information processing and communication in the nervous system. They're found in almost every type of animal, except sponges. The general structure of a neuron is as follows: There is a main body called the *soma*, which as a group of small branching appendages called *dendrites*. There is also a long fiber called an *axon* that stretches out from the neuron. Down the axon are some more branches called *synaptic terminals*, which touch the dendrites of other neurons.
- ▶ The system of neurons works as follows: The dendrites pick up chemical signals called *neurotransmitters* from other neurons. Some of these signals stimulate the neuron, and others inhibit it, but when the overall effect crosses a threshold called the *action potential*, an electrochemical pulse is produced and travels down the axon, causing the synaptic terminals to send chemical signals to other neurons.
- ▶ The soma, which puts together all the inputs from the dendrites, does a kind of information processing as it determines whether or not a pulse is produced. The axon is a transmission line for the output signal to be carried to other neurons. The axon signal is not electrical, but it does involve changes in electric potential; thus, we can detect it electrically. It is actually a propagating change in the chemical concentrations of sodium and potassium ions across the outer membrane.
- ▶ The information speed in the axon signal is not that fast, perhaps tens of meters per second. This means that neuron signals are a million times slower than electrical signals in wires. If we were to place ourselves at one point along the axon and measure the electric potential over time, it would look like this:



- Each of the spikes represents a pulse going past, and the pulses last only for a couple of milliseconds.
 - The interesting thing is that the details of the amplitude and the shape of the spike do not appear to matter. In fact, those details might change a bit for a signal, from one end of the axon to the other, but as far as the neural information is concerned, a spike is a spike. The information is surprisingly binary: There is a spike, or there is not a spike.
 - The only question is when the spikes occur, and even there, it appears that timing details finer than about 1 millisecond are usually unimportant.
- Therefore, we can regard information transmitted down the axon as a discrete series of bits. Each bit represents 1 millisecond of time. When that millisecond contains a spike, that's a 1; otherwise, we have a 0. Further, because a neuron generally rests for a few milliseconds between pulses, the 1s in this binary string are usually separated by several 0s.
- Now we know what neural information looks like: It's a stream of binary digits, mostly 0s. They are generated at around 1000 bits per second and travel down the axon.



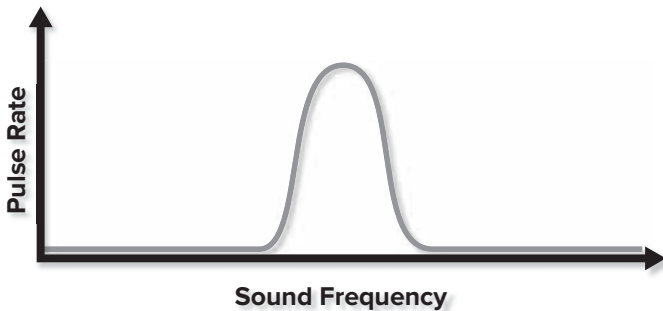
Rate Coding

- ▶ The next question is: What do the bits mean? What is the neural code? The first clue here was found in 1926, when Edgar Adrian and Yngve Zotterman at Cambridge did an experiment on muscle tissue taken from a frog.
 - They monitored the impulses from a single sensory nerve fiber in the muscle while they pulled at the tissue with small weights. They found that the rate of pulses was determined by the amount of pulling force. A larger force produced more frequent spikes. For example, a 1/8-gram weight produced about 20 pulses per second, a 1/2-gram weight produced about 50, and a 3-gram weight produced more than 100.
 - If a steady force was applied, the pulse frequencies decreased over time. This is called *adaptation*—a constant stimulus produces a diminishing sensory response.
 - But the main point remains: In a sensory nerve, the rate of the pulses represents the strength of the stimulus. This idea is called **rate coding**, and it appears to be the basic neural code used by many kinds of neurons, including many within the brain.
- ▶ Rate coding is interesting because individual spikes do not carry the neural message. Instead, the message is encoded in a train of spikes. Thus, the typical time interval required for a neural message must be fairly long. The information is encoded in the number of spikes that occur in that interval. That means that the following two “codewords” are equivalent because they each have nine 1s, that is, nine pulses in the sequence.

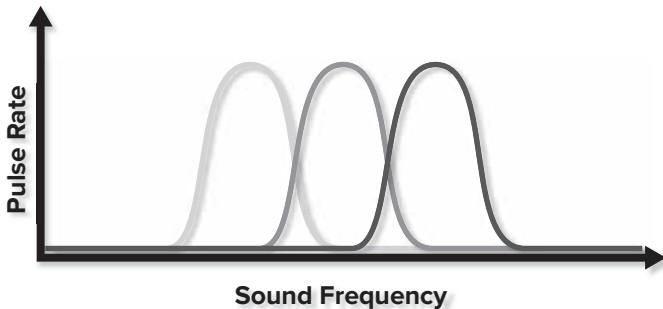
```
000010010000101000010001000000100000100001000...  
001000000100100000100010000001000100000100100...
```

- ▶ How long is the timescale for an actual neural message—the time for a single rate-coded piece of information to be conveyed? It can't be too long, because we know that we can receive and act on sensory data in a fraction of a second. But it can't be too short, because many pulses must be received for the rate to be measured.
 - Let's suppose that this timescale is around 200 milliseconds (ms). The pulse rate for a neuron almost never exceeds 100 pulses per second—10 ms per pulse. Thus, the number of pulses we expect to find in 200 ms is between 0 and 20.
 - Those 21 possibilities correspond to 21 different measured pulse frequencies—the codewords in our rate-based neural code. That number, 21 codewords, is enough for $\log_2(21) = 4.4$ bits per message. At five messages per second, that's 22 bits per second.
 - However, that number is probably too high. The pulses come along somewhat randomly, which means that, by chance, a given interval of time might contain more or fewer pulses. If the expected rate is 9 pulses, the interval might actually contain 6 pulses or 11.
 - Physicists call this variation in the number of discrete random events *shot noise*, and it means that two similar pulse rates—such as 9 pulses per interval versus 8—cannot be reliably distinguished.
 - Because of noise, therefore, the information capacity of the neuron channel is lower: probably 10 bits per second or less.
- ▶ Several factors mitigate this. For one thing, if you have several neurons carrying the same information—producing pulses at the same expected rate—then by combining the signals from all of them, you can make a faster and more accurate determination of that rate. Also, different neurons can carry complementary information about the same thing. We saw this in Lecture 6, when we described hearing.

- Because of their locations within the inner ear, different hair cells are sensitive to different frequencies. Thus, the pulse rate for a particular auditory neuron will be high for one frequency but much lower for higher or lower frequencies. That's called the *tuning curve* for that neuron:

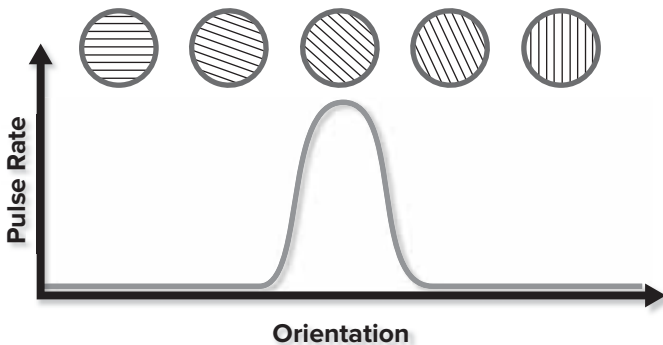


- But different neurons will have different tuning curves that peak at different sound frequencies:



- Even if each neuron provides only a few bits of information, each piece of information is about a different frequency range. The combination of all of them provides a fairly detailed description of the sound.

- ▶ Notice how much of this is parallel to the perceptual coding of audio data, which we discussed in Lecture 6. In the MP3 code, for example, the sound signal for a short interval of time is broken up into frequency bands. Some frequencies are coded with only a few bits. Our auditory system evidently has its own internal MP3 code for sound.
- ▶ Neuron tuning curves show up in some surprising ways. For instance, the visual cortex is the part of the brain that processes visual information. Suppose you are looking at a picture of parallel stripes. There are cells in the primary visual cortex that respond to a particular angle of the stripes.



- Those neurons combine data from many parts of the retina to compute a simple function. Other cells respond to different angles, or to edges, or to movement, or to shapes.
- There are millions of such cells in the visual cortex, and they are a significant part of how we integrate data from our eyes into visual perception.

Temporal Coding

- ▶ The alternative to rate coding is called **temporal coding**. Here, the actual times of particular pulses—or, more precisely, the times between adjacent pulses—make a difference in the message.
- ▶ Below are the two binary pulse sequences we saw earlier. Both have nine pulses. In rate coding, they are equivalent codewords. But because the pulses occur at different times, they are distinct codewords in temporal coding. Therefore, the neural information rate would be higher if neurons use temporal coding.

```
000010010000101000010001000000100000100001000...  
001000000100100000100010000001000100000100100...
```

- ▶ Several pieces of data suggest that precise timing matters in the brain. For instance, suppose you shut your eyes and listen to a sound from different points. You can tell which direction the sound comes from, in part because of different degrees of loudness in your two ears. But experiments prove that most of that sense of sound localization comes from the difference in arrival time of the sound waves at your ears, and that difference is exceedingly small—much less than 1 ms. Thus, tiny timing variations must somehow give rise to information in your brain.
- ▶ Another, more indirect point is this: The nervous system transmits pulse timing information very exactly. The pulses do not get closer together or farther apart as they travel. This is not evidence that this timing information is used in the neural code, but if it isn't being used, why does the nervous system preserve it so well?
 - In addition, if the exact timing of the pulses was literally meaningless, then there would be no reason for different neurons to fire at the same moment; they could produce their pulses independently. They might occasionally produce simultaneous pulses by chance, but that would represent nothing special in the neural information network.

- However, there are many experiments showing that neurons can become synchronized, firing simultaneously much more often than coincidence would allow. Once again, that does not prove that synchronization is part of the neural code, but if not, why does it happen so often?
- ▶ There are many other observations and experiments of the same sort, all tending to show that precise timing makes a difference. For this reason, the current consensus is that temporal coding does play a role in the neural code, at least in certain places. This fact greatly increases our estimate of how much information is being exchanged by neurons within the brain.

Memory

- ▶ Of course, the brain doesn't just process and transmit information; it also stores it, sometimes for decades. This is the phenomenon of memory. The brain appears to have several different memory systems that store different kinds of information, in different ways, for different periods of time.
 - The first of these systems is *sensory memory*, which puts together the fragmentary, moment-by-moment inputs from the visual and auditory systems into unified impressions. It stores a great deal of detail, and it does not appear to be selective. In other words, it does not matter whether or not you are paying attention to something; if that something affects your senses, it goes into sensory memory. But sensory memory is gone after only a moment.
 - The next system is sometimes called *short-term memory*. It is linked to your attention, and it lasts much longer than sensory memory—perhaps 15 seconds. Short-term memory is useful for performing tasks, which is why it is also called *working memory*.
 - Finally, there is *long-term memory*. Information that is stored by this system can be recalled days or years later. Long-term memory storage involves a permanent change to the brain.

- ▶ There appears to be an information filter at each stage in these memory systems. Only some sensory memory information goes into short-term memory. Only some short-term memory information goes into long-term memory. Also, the different timescales over which the systems operate indicate that the physical mechanism is different in each case.
- ▶ Sensory memory is very likely stored as self-sustaining patterns of neural signals within the sensory areas of the brain. These patterns are transitory and are almost immediately changed to new patterns based on new sense data.
- ▶ Short-term memory appears to be the business of the prefrontal cortex, the executive center at the front of the brain. That's where our conscious attention arises, which makes sense, given the connection between attention and short-term memory. How is the information represented there? One leading theory is that patterns of nerve impulses lead to patterns of localized depletion of neurotransmitters.
 - Remember, neurotransmitters are the chemical signals that bridge the gaps (synapses) between one neuron and the next. If a particular neuron has been involved in a great deal of activity, its supply of those molecules will be temporarily reduced. Therefore, the pattern of pulse activity leaves behind a chemical “footprint.” That seems to be the way short-term memory is represented.
 - After a short while, the cells get resupplied. Thus, unless it is renewed by additional activity, that memory trace fades away.
- ▶ Long-term memories of events in our lives or of abstract concepts seem to rely on the hippocampus, a pair of curved structures deep inside the brain. The hippocampus contains a large number of neurons with many connections, and it is linked to the prefrontal cortex and many other parts of the brain.
 - Information is represented in the connections between neurons. Like muscles, those connections are strengthened with use; that process is called **long-term potentiation**. Thus, repeated activation of a given set of connections will cause them to be permanently reinforced—that's memory.

- We know for sure that this happens in the hippocampus. Some memories are actually stored there, such as our memories of spatial information. Other memories appear to be located somewhere else, but the hippocampus assists their formation.
- The hippocampus does not seem to be involved in storing other kinds of long-term data, such as emotional responses or how to ride a bicycle. But this information is also encoded in the strengths of synaptic connections in various parts of the brain. When we recall those memories, we use those connections.
- But notice, that's how we form memories in the first place. Thus, in our memory system, the information is read using the same process by which it was written. Every time we access a memory, we are rewriting it, tracing the pen over the letters on the page. In that process, sometimes the memory information can be altered. This is why it is possible, even fairly common, to have false memories.
- ▶ How many memories can we store? What is the information capacity, in bits, of our memory system? The psychologist Thomas Landauer at the University of Colorado did some top-down experiments on memory in human volunteers.
 - He exposed the volunteers to various kinds of information, then much later asked them some true-false questions about it. Based on his experiments, Landauer concluded that humans assimilate information into long-term memory at a rate of about 2 bits per second. That doesn't sound like much.
 - Thirty years equals 1 billion seconds; thus, a person might have 2 billion seconds of waking life. According to Landauer's estimate, that's only 4 billion bits of memory data—less information than is stored on a single audio CD! It seems certain that Landauer's estimate is much too low.
 - At the other extreme, von Neumann made a wildly generous estimate of memory capacity. He accepted the idea, common in his day, that we actually never forget anything. Thus, he calculated

that a complete record of the brain's neural impulses for a lifetime would constitute about 3×10^{20} bits of information. Of course, we now know that our memory is not just a data recorder for the whole brain. Memories are constructed, and only a portion of our experience is stored.

- ▶ A better answer than either of these would be a bottom-up estimate, based on the number of synaptic connections in the brain; those connections are the basic underlying mechanism for memory.
 - There are about 10^{10} neurons, each of which connects to around 1000 other neurons. That's at least 10^{13} connections—a terabyte of data. But the varying arrangements and strengths of those connections multiply that number considerably. One recent estimate of the information content of our brain network is 2.5 petabytes—more than 20 million billion bits.
 - That is an enormous information capacity, and in fact, there is no evidence that even in a long human lifetime, we ever approach any limit to our ability to record new information in the memory system. The memory difficulties some people experience always have particular physiological causes; they are never about the underlying information capacity.
- ▶ There are still a great many unanswered questions about this magnificent information system of ours, the brain. How can we bridge the gap between the top-down and bottom-up approaches—the world of perception and experience and external behavior on the one hand and the intricacies of neural codes and long-term potentiation on the other? With models and experiments and ideas drawn from the science of information, the next couple of decades of neuroscience research may tell.

TERMS

long-term potentiation: The process by which connections between neurons are strengthened by use. This is the physical basis for long-term memory. The fact that the process of “reading” a memory is the same as the process of “writing” a memory means that memory is not a simple record of events; our memories can be changed over time.

neuron: The basic unit of the brain for information processing and communication. Each neuron consists of a main body called the *soma* and a long fiber called an *axon*. Connections from other neurons induce the soma to produce nerve impulses, which are transmitted down the axon and influence other neurons connected at its end.

rate coding: A neural code in which the information is represented by the rate of the electrochemical pulses that travel down the axon. It has been known since the 1920s that at least some neural information is encoded in this way. Because the pulses occur somewhat at random, the capacity of this kind of code is limited by shot noise to a few bits per second. See also **temporal coding**.

temporal coding: A neural code in which the information is represented by the exact timing of the electrochemical pulses that travel down the axon. Temporal coding allows a greater information capacity than simple **rate coding**. Several lines of indirect evidence suggest that this is part of the neural code in at least some parts of the brain.

READINGS

Pierce, *An Introduction to Information Theory*, chapter 12.

Siegfried, *The Bit and the Pendulum*, chapters 6–7.

QUESTIONS

- 1 As we saw in this lecture, one estimate for the memory capacity of the human brain is 2.5 petabytes. As mentioned in Lecture 5, at the present time a small 1-terabyte portable drive costs around \$50 and weighs around half a pound. If you wanted to “back up” your brain onto a number of such drives, how expensive (and heavy) would the storage system be?
- 2 In an actual experiment, human subjects were asked to read randomized lists of words. In one test, the words were drawn from a vocabulary of 256 words. In another, the vocabulary was 5000 words. The subjects could read 3.8 words/second in the first case and 2.7 words/second in the second. From this, estimate the word-reading capacity of the subjects in bits per second.

When the concept of entropy was originally discovered in the context of thermodynamics, it seemed to have nothing to do with information. It was simply a way of determining whether or not an energy transformation was possible. However, when we look at the story of its discovery through our “information-colored glasses,” we can see entropy for what it really is—because thermodynamics is not just a practical and profound branch of physics; it is also a place where the laws of nature and the laws of information meet.

The First Law of Thermodynamics

- ▶ The first big idea in thermodynamics is the familiar idea of energy. Every separate part of the Universe—called a *system*—has a certain energy content, which we denote by E . The physics unit of energy is the joule. Energy can change forms and be transferred from one system to another, but the total amount of it remains constant. In other words, energy is conserved. That’s the first law of thermodynamics.
- ▶ Systems can exchange energy in two distinct ways. The first way is called *work*, W . This is energy transfer that is associated with a force acting through a distance, such as the force of gravity acting on a baseball after you drop it. Less obviously, work is also done when you charge up an electric battery or when a material becomes magnetized.
- ▶ The second way that energy is transferred is called *heat*, Q . If you put two bodies at different temperatures next to each other, there is a heat flow from the hotter one to the colder one. That energy transfer isn’t work because the objects are not exerting forces or being moved.

- ▶ According to the first law, a system's energy, E , cannot just go up or down on its own. There must be some kind of energy transfer to or from the system. Thus, the total change in the system's energy— ΔE —is just heat plus work: $Q + W$. Notice that Q and W could each be positive or negative, depending on whether the energy is flowing into or out of the system.
- ▶ Any sort of energy transformation or exchange aligns with the first law. It does not matter, for instance, whether heat flows from hot to cold or cold to hot; either direction obeys the law. But in the real world, heat only flows from hot to cold, never the reverse.
 - In addition, we can always transform work into heat. If you rub your hands together, they warm up. Friction turns muscular work into heat. But the reverse is not always possible; we cannot always turn heat into work.
 - To a degree, a steam engine is an example of turning heat into work. Heat from a fire boils water, and the steam pushes a piston. But not all the heat is turned into work. Some of it is released into the cooler surroundings as *waste heat*. And try as we might, we cannot build an engine that turns heat into work with 100% efficiency. Thus, there is another principle at work besides the first law.

The Second Law of Thermodynamics

- ▶ The second law of thermodynamics was discovered by the German physicist Rudolf Clausius in the mid-19th century. To express his law, Clausius introduced a strange new quantity called *entropy*— S . The word derives from the Greek word *tropos*, meaning “transformation.”
- ▶ Entropy is a property of a system, like its mass or energy content. It can change as the system undergoes change and moves from one system to another when heat is transferred. According to Clausius's second law, the total entropy can never decrease.

- ▶ If an amount of heat, Q , is transferred to a system at temperature T , Clausius says that its entropy changes by $\Delta S = Q/T$. The units of entropy are energy divided by temperature: joules/kelvin.
 - Suppose some heat flows from system 1 to system 2—from the hot coffee to the cool cup. The amount of heat that flows is Q . Because system 1 loses heat, its entropy decreases by Q/T_1 . Because system 2 gains heat, its entropy increases by Q/T_2 . The total change in entropy is $\Delta S = Q/T_2 - Q/T_1$.
 - The second law tells us that this must be positive; total entropy cannot decrease. Thus, the first term must be greater than the second, which means $T_2 < T_1$. The heat always flows from the higher temperature (T_1) to the lower one (T_2).
- ▶ Returning to the steam engine, heat flows from the boiler; thus, the boiler's entropy decreases. The engine produces some work, but work doesn't affect the entropy. To satisfy the second law, therefore, the engine must expel some waste heat to raise the entropy of the surroundings. That's why it cannot be 100% efficient.
- ▶ We do not directly perceive or experience entropy. By doing experiments and seeing which processes happen and which do not, we can measure it and learn that certain factors make the entropy of a system higher or lower. For example, a gas has a higher entropy than a solid. The gas entropy is higher when it occupies a larger volume. A system has a higher entropy when it is hot rather than cold and when its parts all have the same temperature. Yet none of these observations answers the question: What is entropy?

Defining *Entropy*

- ▶ That question was answered by the Austrian physicist Ludwig Boltzmann about a dozen years after Clausius introduced the entropy concept. The real meaning of Boltzmann's discovery is that entropy is information.

- ▶ Everything we see around us is macroscopic. It is composed of huge numbers of microscopic particles—atoms and molecules—and all of those particles are in ceaseless motion. Although we often think of that motion as random, that isn't quite right. Atoms move and exert forces on each other according to the usual laws of physics. But there are so many atoms and their motions and interactions are so incredibly complicated that we know almost nothing about the details of which atom goes where.
- ▶ In fact, for almost any macroscopic system, we know only a few bits of information, such as the mass, volume, temperature, and so on. That small amount of large-scale information is called the **macrostate** of the system. We do not know the **microstate**—the exact situation of every individual atom.
- ▶ There are many, many, possible microstates that are consistent with the macrostate. Let's call that number of possible microstates M —a mind-bogglingly huge number. Because there are M possible microstates, and we don't know which one we have, we are missing $\log_2(M)$ bits of microstate information. That's the Shannon entropy, H , of the microstate message. That's how many binary digits we would need to record all the details about all the atoms in the system.
- ▶ Here is Boltzmann's discovery in information language: The thermodynamic entropy of Clausius, S , is simply a constant (k_2) multiplied by the microstate entropy: $S = k_2 \log_2(M) = k_2 H$. That's our version of Boltzmann's formula. Thermodynamic entropy and information entropy are really the same. What is k_2 ? It's the thermodynamic value of 1 bit of missing microstate information. That value is very small, about 10^{-23} joules/kelvin per bit.
 - Let's look at a simple example. Consider a jar containing a liter of air at room temperature. The thermodynamic entropy of the gas turns out to be around 6 joules/kelvin.
 - How many bits of information do we lack about the microstate of the gas? According to Shannon, the information in bits is $H = S/k_2$, or 6×10^{23} bits. That vast amount of information is how much we do not know about what is in the jar.

- ▶ Boltzmann, of course, did not think about entropy as microstate information. He regarded entropy as a measure of “mixed-ness” or disorder. The larger the number of possible microstates, the more mixed-up and disorderly the system is. But the information view is more fundamental.
- ▶ Suppose we take a liter of gas and expand it to twice the volume, 2 liters, while keeping it at the same temperature. Clausius would calculate its change in entropy as follows:
 - We could make the change by slowly expanding the gas from 1 liter to 2 liters. Ordinarily, a gas would cool as it expands, so we should add some heat to maintain the same temperature. Adding heat adds entropy. The change in entropy is just Q/T .
 - Knowing some details about how gases behave, we can determine the answer. We have to add about 90 joules of heat as the gas expands, and the temperature is 300 kelvins. The change in entropy is $90/300 = 0.3$ joules/kelvin. Expanding the air increases its entropy about 5%.
- ▶ Now, let’s look at the same problem from an information science perspective: It would take a large number of bits to describe the microstate of the gas molecules in a 1-liter volume. How many more bits would it take to describe them if they occupied a 2-liter volume?
 - We can think of the larger volume as being made up of two 1-liter halves. To describe where a molecule is, we first have to tell which half it’s in. That’s 1 bit of information.
 - But once we’ve narrowed it down to a 1-liter volume, we lack exactly the same information about that molecule that we lacked before. We still need to specify where it is within a 1-liter volume, how it is moving, and so on.
 - In other words, by doubling the volume, we have added exactly 1 additional bit of microstate information for each molecule. If we have N molecules, then ΔH , the change in missing microstate information, is N bits. Therefore, the thermodynamic entropy has increased by $\Delta S = k_2(N \text{ bits})$.

- It turns out that in 1 liter of air, there are about 3×10^{22} molecules; that works out to a thermodynamic entropy change of 0.3 joules/kelvin.
- That is exactly the right answer, the same answer that we found by keeping track of the heat added as the gas expands. But thinking about information makes the meaning obvious. If we double the volume, each air molecule has twice as many places it can be. That means we lack 1 additional bit of microstate information for each molecule.
- ▶ Here's a related example: Suppose the gas is composed of two kinds of molecules in equal amounts, say, 50% nitrogen and 50% oxygen.
 - In one situation, the two gases are separate, each in half the volume, separated by a thin wall. In a second situation, the two gases are mixed together throughout the volume. The mixed-up situation has a higher entropy.
 - The difference is called the *entropy of mixing*, and it is easy to see why there should be a difference. If the two gases are separate, we know 1 more bit of information about each molecule—that each nitrogen molecule is in one half and each oxygen molecule is in the other half. But when the gases are mixed, any of the molecules could be anywhere. Thus, with N molecules altogether, that's an N -bit difference in microstate information, or in thermodynamic entropy terms, $k_2(N)$.
- ▶ You will notice that the Boltzmann formula is related to the Hartley-Shannon formula for information entropy—the logarithm of the number of possibilities. This assumes, as Boltzmann did, that all the possible microstates are equally likely, but there are situations in which some microstates are more probable than others. For instance, because of gravity, one nitrogen molecule in a room is slightly more likely to be near the floor than near the ceiling. Thus, not all microstates are equally likely.

- Boltzmann's formula for entropy was extended to these situations by the American physicist Josiah Willard Gibbs. He found that if the microstate m has probability $p(m)$, then the thermodynamic entropy is:

$$S = k_2 \sum_m p(m) \log_2 \left(\frac{1}{p(m)} \right).$$

- When Shannon, thinking about coding and communication, discovered his measure of information, he recognized the link to Gibbs's formula. That's why he used the term *entropy*.

The Principle of Microstate Information

- ▶ The importance of entropy for physics is that it tells us which processes are possible and which ones are not. The total entropy can stay the same or increase but never decrease. What does that have to do with microscopic information?
- ▶ To understand the connection, begin by imagining the collection of all conceivable microstates of a system. That forms what physicists call the system's *phase space*. Each point in the space is a microstate of the system. Two different points are two microstates that differ in some way, such as the arrangement of atoms in the space, how they are moving around, and so on. Even one small difference in a single atom will give us two different microstates. When we talk about microstate information, we're talking about detailed coordinates—exactly where the system is located in phase space.
- ▶ What is a macrostate in this conception? The macrostate of the system provides some information about the microstate, but it leaves out a great deal of detail. If phase space is like a map of the United States, a macrostate is like Texas. Knowing that you are in Texas says something about where you are, but there are still a great many different specific points in Texas where you could be.

- ▶ Thus, in our conception of phase space, a macrostate is a region containing many microstate points. Some macrostates are smaller, containing fewer microstates, and others are larger, containing more microstates. The larger regions have a greater entropy, because you'd need more information to locate the actual microstate within that region. Thermodynamic entropy is k_2 multiplied by the information we lack about the microstate.
- ▶ Over time, the microstate of a system changes. For instance, in a jar of air, the molecules move and collide with each other. The macrostate might not change, but the microstate continually changes in complicated ways. And this is true whether we think of the gas molecules as little balls that fly around and exert forces on each other, acting according to Newton's laws of motion, or as quantum mechanical particles that propagate and interact as waves, according to quantum laws. It is also possible that changes in the microstate lead to changes in the macrostate, but the laws that govern those changes are still the microstate laws.
- ▶ However, there is a remarkable and important fact about the laws that govern how microstates change, known as the *principle of microstate information*: As microstates change, microstate information is preserved.
 - As we've said, information has to do with the distinction between things. Imagine two microstates that differ, if only in the location of one particle. They represent two possible ways the system could be at a given moment.
 - The laws of physics cause the microstates to change as the particles move and interact. Each of the two microstates wanders around in phase space, following those laws. But the principle of microstate information tells us that they never wander to the same place. They must always end up as two different microstates, preserving the distinction between them.
- ▶ Suppose a system starts out in a macrostate—Delaware. We do not know the microstate of the system; it could be any point in Delaware. Over time, the microstates wander around according to the microscopic

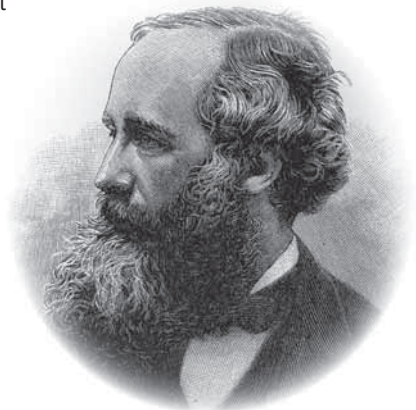
laws. The principle of microstate information tells us that they occupy a region exactly the same size as the one they started in—a region containing the same number of microstates.

- ▶ Imagine now that a system starts out in one macrostate and winds up in another. For instance, we might start out with separate oxygen and nitrogen gases, but then we allow them to become mixed. In phase space, the system starts out in one region, then ends up in another. Whatever happens does so because of the laws governing the microstates.
 - All the different microstates in the initial region end up as different microstates in the final region. Therefore, the final macrostate—the region at the end of the process—must be at least as large as the initial macrostate, and it might be larger. The points in Delaware could all end up in Texas, but Texas could not fit into Delaware.
 - Because of the principle of microstate information, the final macrostate can be no smaller than the initial one. And that means the entropy of the macrostate cannot decrease. That’s the second law!
- ▶ This is an amazing connection between information and physics. Because of the principle of microstate information, distinct microstates remain distinct as the system evolves. But we don’t see microstates; we see only the big picture, the macrostate. Yet the principle of microstate information does have a consequence that we can see: A larger macrostate can never be squeezed into a smaller one. Total entropy can never decrease.
- ▶ If the principle of microstate information were not true, if different microstates could evolve into the same microstate later on, then the second law would not be true either. There would be no problem fitting Texas into Delaware, because many different points in Texas could end up at the same point in Delaware. Entropy could decrease.

- ▶ What we have called the principle of microstate information goes by many other names in physics books, including Liouville's theorem, overall unitarity, or microscopic reversibility. But it is really a law about information in the physical world. It tells us that at the fundamental level, the Universe does not discard information. And the second law of thermodynamics, the law of increasing entropy, emerges from that fact.

Maxwell's Demon

- ▶ The first person to glimpse a connection between entropy and information was the brilliant Scottish physicist James Clerk Maxwell. He thought that the second law depended on the fact that we can see only the big picture. We have access only to macroscopic information. Individual atoms and molecules are much smaller than we are. We do not see them, and we cannot manipulate them one by one. Thus, Maxwell believed that the second law was really an expression of our own limitations.
- ▶ To illustrate this, he devised a famous thought experiment. He imagined a tiny being that came to be called **Maxwell's demon**. The demon is so small that it can perceive and act on individual molecules.
- ▶ Imagine we have a container of gas that is a mixture of oxygen and nitrogen. There is a wall in the middle with a small opening, but the gases are mixed on both sides. Now, we station the demon at the



JAMES CLERK MAXWELL

opening and allow it to open or close a trapdoor. The demon plays the following game:

- If a nitrogen molecule approaches from one side, the demon lets it through. If it approaches from the other side, the demon closes the door and it bounces back. The demon does the same for oxygen molecules, except it lets them through in the other direction.
 - After a long while, the demon's game will sort the molecules out: All the oxygen will be on one side, and all the nitrogen on the other. The gas goes from mixed to separate; the entropy decreases. In other words, the action of the demon can contradict the second law.
- The demon can violate the second law in other ways by playing the game differently. If it lets every type of molecule through in one direction but not the other, it can eventually cause the whole gas to occupy a smaller volume. If the demon sorts the molecules by their speed, so that faster molecules end up on one side and slower ones on the other, it can produce a temperature difference where none existed before. In any of these ways, the demon can engineer a decrease in the entropy of the gas.
- In other words, Maxwell seems to say that if we could have access to the microscopic information that we usually lack, then the second law could no longer limit us. Any energy transformation would be possible. We could make heat flow from cold to hot; we could turn heat into work with 100% efficiency.
- Was Maxwell right? Is the second law really about us and our limitations, rather than about nature? For 100 years, one physicist after another tackled this puzzle. Most of them believed that Maxwell had left something out of his thought experiment. Perhaps if we analyzed carefully how the demon might actually work, then we would discover that it could not reduce entropy, and the second law would be valid after all. Yet despite repeated analysis, Maxwell's demon proved difficult to exorcise in a definitive way.

- To solve the puzzle of Maxwell's demon, we must delve even more deeply into the link between information and physics. A real demon would have to be a physical system that interacted with its surroundings—a kind of robot or computer. What does it mean for such a system to acquire, store, process, and use information? How does it affect the entropy of its environment as it operates? Only by understanding these questions can we see the demon for what it is and learn from it profound new lessons about entropy and the physics of information.

TERMS

macrostate: The condition of a large-scale thermodynamic system, characterized by a few large-scale variables, such as energy, volume, temperature, and so on. See also **microstate**.

Maxwell's demon: A thought experiment in thermodynamics by James Clerk Maxwell, in which a demon acquires and uses detailed molecule-by-molecule information to reduce the entropy of a system, such as a container of gas. However, the second law of thermodynamics is not contradicted. Once the demon erases the vast quantities of data it has acquired (a necessary step for it to continue to operate), it will, by Landauer's principle, create enough waste heat in its environment to ensure that the total entropy will not decrease.

microstate: The detailed condition of a thermodynamic system, characterized by the exact state of every molecule it contains. This is a vast amount of information, even for a relatively small system, and we generally possess almost none of it. Thermodynamic entropy is essentially the amount of microstate information we lack. By the principle of microstate information, the distinction between microstates is preserved as the system evolves in time.

READING

Atkins, *The Laws of Thermodynamics*.

QUESTIONS

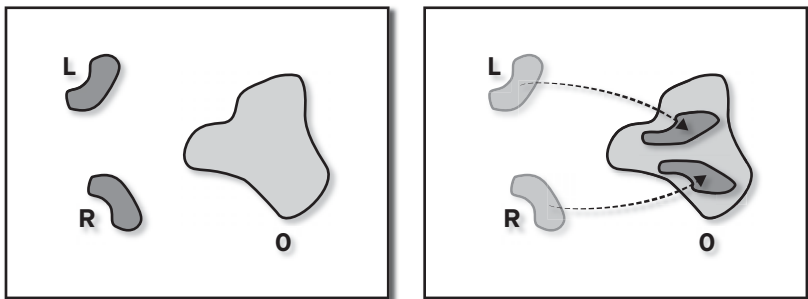
- 1** Consider a liter of air in a jar in your laboratory. Estimate your complete knowledge about the gas, including your knowledge of its history, its visual appearance, and so on. Compare this to the Shannon entropy of the microstate information you lack.
- 2** One gram of water contains about 3.3×10^{22} water molecules. It takes about 2000 J of energy to boil a gram of water at 373 K (100° C). How many bits of information are lost per water molecule when it boils?
- 3** Some physicists have said that the principle of microstate information (according to which microstate evolution is never lost) seems incompatible with the second law of thermodynamics (according to which some macrostate changes are irreversible). Based on the discussion in this lecture, how would you respond?

The second law of thermodynamics that Clausius discovered a century and a half ago turns out to be exactly equivalent to the following statement: No process can have as its only result the erasure of information. In this lecture, we'll explore why the erasure of information is so important and what it has to do with the second law.

Preservation and Loss of Information

- ▶ According to the principle of microstate information, the laws of physics for atoms and molecules preserve information about their state. If we imagine two different initial microstates for a system—states that differ in some way, such as in the location of the molecules or how they are moving—the states will evolve over time into two different final microstates. Yet in the everyday world, in which we only see macrostates, things work differently. Macroscopic information is not preserved; it can be lost.
- ▶ Consider a book sitting on a table. It is at rest now, but perhaps it was moving a minute ago. It might have been sliding across from one side to the other, but friction slowed it to a stop. Its kinetic energy (energy of motion) was converted to heat. Two different initial macrostates—a book moving this way or a book moving that way—both lead to the same final macrostate: a slightly warmer book sitting at rest. That macrostate has “forgotten” which microstate it came from. How did that happen?
- ▶ In phase space, the abstract space in which each point is a microstate, macrostates correspond to large regions containing many microstates. We have a region we'll call L that corresponds to the book moving left and another called R that corresponds to the book moving right. There is also a macrostate region called O corresponding to the book at rest.

- The microstates move around in phase space. Microstates that start out in either the L or R regions end up in the O region. That can happen because the final region O is larger than the other two. There are more microstates consistent with that macroscopic information, and therefore, the entropy is higher.



- We can learn many things from this. In the microscopic world of individual atoms and molecules, there is no friction. Friction is a big-picture thing, a phenomenon of systems with trillions of molecules. Friction can cause the loss of macrostate information—the distinction between a book moving to the left or to the right—and it can produce an increase in entropy. As we will see, those two effects are inextricably linked.

Exorcising Maxwell's Demon

- Maxwell's famous demon is a hypothetical entity that uses microscopic information to control the behavior of a system. The classic version has the demon operate a tiny trapdoor between two containers of gas. In this way, it can achieve effects that at least appear to violate the second law and reduce the system's entropy.
- Over the years, many physicists have tried to find a loophole. They thought about the demon, not as a supernatural entity but as a physical being of some kind, operating according to the laws of physics. By

carefully considering how such a demon would function, they hoped to show that it could not work as Maxwell supposed. They tried to “exorcise” the demon from thermodynamics.

- ▶ The Hungarian Leo Szilard and, later on, the French Nobel laureate Leon Brillouin analyzed the demon as a system that acquires and uses information. Starting from quantum physics, they argued that the demon must expend energy to acquire its information about the gas molecules. Perhaps it has to bounce photons off of them, which takes energy. The energy is dissipated as heat, leading to an increase in the entropy of the surroundings. This offsets the decrease in entropy achieved by the demon and, overall, vindicates the second law.
- ▶ Szilard and Brillouin were almost right. A real Maxwell’s demon—a physical entity that makes use of microstate information—cannot, in the end, violate the second law. But Szilard and Brillouin were wrong about the reason why this is so. It isn’t the acquisition of information that creates extra entropy; it’s what happens to the information afterward.

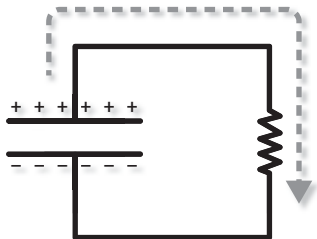
Landauer’s Principle

- ▶ One of the people who read Brillouin’s work was a physicist at IBM named Rolf Landauer. He wondered: What do the fundamental laws of physics tell us about systems that process information? What principles govern the physics of computers?
 - Landauer’s own research was about how electrons move inside materials—the basic physics, in other words, of modern electronic computers. But his question was wider.
 - Regardless of the technical details, what can we say in general about computation as a physical process?
- ▶ We know that computers produce waste heat as they operate. They produce so much heat that getting rid of it is a real challenge in computer design. But Landauer asked: At a fundamental level, why does

computing produce any waste heat at all? One answer is speed; going fast produces more heat.

- Let's consider an oversimplified picture of 1 bit in a computer. The bit is essentially a capacitor, which is an arrangement of two metal plates next to each other. Some electrons can be shifted from one plate to the other. That makes one plate positively charged and the other negatively charged. When one side has the positive charge, the bit is a 0. When the other side has the positive charge, it's a 1.

- Now suppose we want to change the bit from 0 to 1. We have to run the electrons quickly around from one side to the other. We need a wire for that, and the wire has some electrical resistance—electron friction (represented by the zig-zag line in the diagram). Friction produces waste heat.

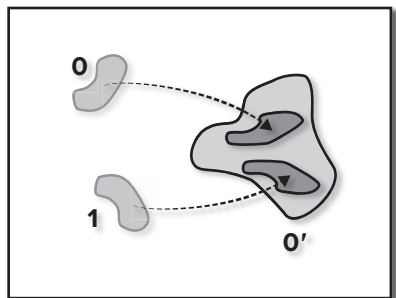
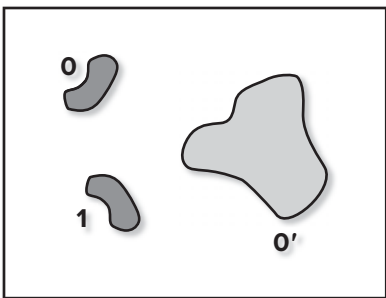


- Next suppose we slow things down. We shift the charge from one side to the other only 1/10 as rapidly. The electric current in the wire is only 1/10 as large, but it flows for 10 times as long. Here's the interesting point: Heat dissipation is proportional to the square of the electric current. Thus, waste heat is now generated at only 1/100 the rate.

- Everything takes 10 times as much time, of course, but that still means that the total heat generated is only 1/10 as great as before. We can do the same operation—moving the charge to change a 0 to a 1—with only 1/10 the waste heat.

- Landauer found that even if you design a computer perfectly and are content to run it very slowly, there would still be some waste heat. Entropy would be increased. The process would be thermodynamically irreversible, and the reason for that is that computation is logically irreversible.

- Suppose we have a computer, and in its memory are two numbers, 3 and 4. The computer calculates their sum and stores the sum in the same memory location. Before, we had 3 and 4; afterward, we have 7. That computation is logically irreversible.
 - From the final memory state, 7, we cannot deduce the initial memory state. It might have been 3 and 4, or it might have been 5 and 2. The computer has “forgotten” exactly how it started out.
- Whenever that happens, Landauer said, there is always an increase in entropy somewhere. If nothing else, the computer must release waste heat into its surroundings. He illustrated this by considering the simplest possible irreversible bit operation: erasure.
- When we erase 1 bit in memory, we start out with a bit that may be either 0 or 1, then reset it to a particular value, say 0. The information is not stored anywhere else. The computer cannot unerase it. Once it is erased, the bit has been lost.
 - The computer and its surroundings form a physical system with a phase space. Each microstate of the system corresponds to a point in that space. Initially, the memory bit of the computer is either 0 or 1. Each of these corresponds to a whole region in phase space, a macrostate. That’s because there are many different detailed arrangements of atoms of the computer consistent with either bit value. Thus, the microstate starts out in one of two possible regions, which we will label 0 and 1. Each region contains M microstates.



- Afterward, the memory is 0, but there might be some other changes in the computer, giving us another macrostate, which we'll call $0'$. In the process of erasing the bit, all of the possible initial microstates wind up there. But by the principle of microstate information, that new region must be at least as large as the first two combined. If M' is the number of microstates in region $0'$, then $M' \geq 2M$.
- Note that the memory bit—the physical thing that stores the 0 or 1—might be very tiny. It might even be a single atom. We've referred to 0 and 1 as distinct macrostates, but the physical difference between them might not be very “macro.” That doesn't matter; the important thing is that they are distinct.
 - To capture this idea, we'll use the word **eidostate** (Greek: *eidos* = “to see”). The eidostate encompasses all the information that is available to the computer—all of the data that the computer can currently see and use. That includes macroscopic information, such as its temperature or battery level, and it includes the 0s and 1s in memory.
 - Therefore, two different memory records will correspond to different eidostates, even if the memory units happen to be microscopic.
- The computer, like us, lacks information—perhaps quite a bit—about which atom is where and what they are all doing. Each different eidostate is consistent with a vast number of possible microstates. It's a region in phase space. Thus, thermodynamic entropy is just $k_2 \times \log_2$ of that number—the number of bits of microstate information that the computer lacks.
 - The eidostates 0 and 1 (each containing M microstates) both evolve into the same eidostate $0'$ (containing at least $2M$ microstates).
 - It is exactly the same situation as when we slid the book across the table. Because the system forgets where it started out, the final eidostate must be larger. It must have a greater entropy. We can calculate the change in entropy as follows: $\Delta S = S_{\text{final}} - S_{\text{initial}}$. S_{final} is $k_2 \log_2(M')$, which is at least as great as $k_2 \log_2(2M)$. Therefore, ΔS

is at least $k_2 \log_2(2M) - \log_2(M)$. The difference of two logarithms is just the logarithm of the ratio. Thus, we have $k_2 \log_2(2M/M)$ or $k_2 \log_2(2)$. The \log_2 of 2 is just 1.

- Therefore, the entropy of the whole system—computer plus surroundings—must have increased by at least k_2 – around 10^{-23} joules/kelvin per bit. That entropy increase is Q/T , where Q is the waste heat and T is the ambient temperature. Therefore, the waste heat Q is at least $k_2 T$.
- At room temperature, 300 k, that isn't much: 3×10^{-21} joules per bit erased. When your own computer erases information, it generates much more waste heat than this. But this is the absolute minimum, and this fact is known as *Landauer's principle*: Erasing 1 bit of information produces waste heat of at least $k_2 T$.

Reversible Computation

- ▶ Landauer proposed his principle back in 1961. At the time, he did not fully understand which computer operations had to produce waste heat. Years later, another IBM scientist, Charles Bennett, took the next step. He realized that every computer operation could, in principle, be made logically reversible. A computer could operate, in principle, with no waste heat at all. Erasure produces entropy, but computation does not require erasure.
- ▶ To see how a reversible computer might be designed, let's go back to one of Shannon's ideas: that any computation can be done by a complex network of 1-bit messages connecting simple Boolean logic gates. Unfortunately, those logic gates are not particularly reversible. Consider the AND operation, represented by a small wedge, as shown here.

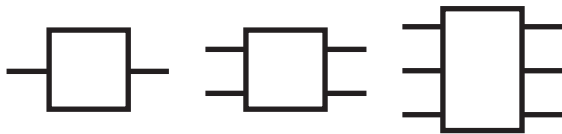


- Here, the AND gate is a small box with two input lines and one output line. Each line carries a bit, either 0 or 1. The output bit is 1 if the input bits are both 1, but otherwise, the output is 0. That is not reversible at all; we cannot always reconstruct the input from the output. If the output is 0, the input bits might be 00, 01, or 10. The AND gate forgets its input. The same is true of the OR gate and the XOR gate. Each of them destroys some information about the input bits.

- One of the basic logic gates is reversible: the NOT gate. It has one input and one output. If the input is 0, the output is 1 and vice versa. That's a reversible operation; just apply a second NOT gate, and the original bit value is perfectly restored.



- If we want to do a reversible computation, we need to use reversible logic gates. These are designed by the *rules of reversible information processing*. The first of these rules is that the logic gate must have as many bits of output as there are bits of input. If there is 1 input bit, there is 1 output bit, like the NOT gate. But if a gate has 2 input bits, it must also have 2 output bits. If it has 3 input bits, it must have 3 output bits.



- The second rule is that the logic gate must preserve the input information, and two distinct inputs must lead to two distinct outputs. Let's illustrate this rule by describing a useful 2-bit reversible logic gate: the controlled-NOT gate, or CNOT.
 - The 2 bits play different roles. One is the control bit, c . Nothing happens to the control bit; it stays unchanged in the output. The second bit is called the target bit, t . If the control bit is 0, then the

target bit is left alone. If the control bit is 1, then the target bit is negated; 0 turns into 1 and vice versa. What we're really doing is replacing the target bit t with $c \oplus t$.



- We can see that the gate has the right number of output bits; thus, the first rule is fulfilled. To verify the second rule, we can make a table of input and output values.

c	t	$c' = c$	$t' = c \oplus t$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

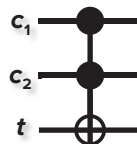
- The output bits, considered together, are different in each row. Thus, no two different inputs lead to the same output. The gate preserves the input information.
- The CNOT gate is so useful that there is special symbol for it in diagrams, as shown here.



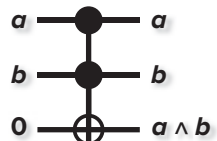
- If the target bit starts out as 0—a fixed constant—and the control bit is 0, both bits end up 0. If the control bit is 1, both bits end up 1. Either way, we end up with two copies of the control bit. This is how we can copy information in a reversible way. We need to start out with a target bit of 0—some “blank memory” in which to create the copy. Then, we use the CNOT operation.

- ▶ We can also “uncopy”—the reverse of copying. We start out with two copies of the same bit, either 00 or 11. If we apply the CNOT, the control bit is unchanged but the target bit is always set to 0.
 - Notice that this does not count as erasing information; it is really just data compression. We start out with only 1 bit of information, stored redundantly in two binary digits. At the end, we still have the same bit, but now it’s stored more compactly, in one binary digit.
 - Consequently, we have freed up some blank memory; the other bit is now 0, and that bit is available for storing some other information.
- ▶ In ordinary, nonreversible Boolean gates, 2-bit gates were enough. We can build any complex mathematical operation out of a network of those. In reversible logic, it turns out that we need at least one 3-bit gate. There are several choices.

- The Toffoli gate is sometimes called a controlled-controlled-NOT, or CCNOT. It has two control bits, c_1 and c_2 . They don’t change. The target bit is negated only if both c_1 and c_2 are equal to 1. We can diagram that like the CNOT gate but with two control bits.



- We can use the Toffoli gate to construct reversible versions of the standard Boolean gates. For instance, suppose we arrange that the target bit input is always 0. That bit changes to 1 only if both control bit inputs are equal to 1. The target bit is now the AND of the other two bits, and we have an AND gate! It’s reversible because it retains the two input bits; thus, no information has been lost.



- We can also make reversible versions of the OR and XOR gates.

Building Reversible Gates

- ▶ Can such reversible gates actually be built? Toffoli, along with Edward Fredkin of MIT, proposed an idealized model. It is less of a practical blueprint than a proof-of-concept. The idea is to do our computing with billiard balls.
- ▶ The balls move along various possible paths, corresponding to the bits in our diagram. If there is no ball on one path, that's a 0. If there is a ball, that's a 1. The balls bounce off of fixed obstacles to change direction, and logic operations are performed by precisely timed collisions between the balls. Because there is no friction and the collisions are perfectly elastic, everything is reversible. Send the balls backward on their paths, and the computation would run backward. There is no waste heat.
- ▶ The billiard ball idea sounds more like a vast pinball machine than a practical computer, but it illustrates a point. A real physical computation is actually performed by the laws of physics—in the billiard-ball computer, according to Newton's laws of motion. And that, in the deepest sense, is what a computer is: a piece of nature that we have arranged so that its natural evolution encodes some mathematical calculation.
- ▶ Nowadays, there are more serious proposals for making reversible logic gates, using superconducting circuits and so forth. In addition, Bennett's ideas about reversible computing have inspired engineers to find ways of reducing the heat dissipation of computer operations in regular computers. Computers today dissipate a million times less heat energy per operation than the ones that were around when Bennett first dreamed of reversible computing—a significant increase in efficiency.
- ▶ The Toffoli gate achieves its reversibility by keeping around some extra bits at the output end; this is no accident. Bennett's reversible computer does not have to generate waste heat, but it does produce waste information—extra bits stored in memory that are not part of the final answer and about which we may not care. If we want to eliminate them, we will have to erase them, and that means we will have to pay Landauer's

price, k_2T per bit. That is the only point where waste heat or entropy increase is necessary. Erasure is the only truly irreversible thing you can do to information.

The Landauer Form of the Second Law

- ▶ This all started from thinking about Maxwell's demon. Does the demon provide a loophole in the second law of thermodynamics? Bennett thought about Maxwell's demon as a reversible computer, making it reversible to avoid unnecessary waste heat. As it operates, it acquires information about the molecules in a gas and uses that information to sort the molecules out. The demon does not produce any waste heat, and the entropy of the gas appears to decrease. The second law seems to be contradicted.
- ▶ However—and this is crucial—something else has changed. The demon's memory is now full of the data it acquired about the molecules—a huge amount of information—and the demon cannot continue to operate once its memory is full. In other words, the demon has consumed a resource—blank memory—to accomplish its task.
- ▶ Bennett argued that if we want to unambiguously contradict the second law, the demon must operate in a closed cycle. It must return to its own initial state. That way, it can continue to function. But that means that the demon must eventually erase its memory record. Erasure is not reversible; it must dissipate at least k_2T of heat per bit erased. And that cost of information erasure—and the entropy produced in the surroundings—is exactly enough to save the second law.
- ▶ Szilard and Brillouin were wrong. The problem with the demon is not the cost of acquiring information. A cleverly designed demon can do that without generating any entropy. The problem is when the demon eliminates information. Because of Landauer's principle, that cannot be done for free.

- ▶ This brings us to the new statement of the second law that we saw at the beginning of the lecture: No process can have as its only result the erasure of information. This seems related to Landauer's principle, though it sounds less specific. Both say that the erasure process is always accompanied by some other net change in the world. But the new statement does not specify what the cost might be. Let's call it the Landauer form of the second law.
- ▶ The Landauer form is truly equivalent to the Clausius form of the second law. Each one implies the other. Put another way, any machine that can violate one form of the law could be used to violate the other form.
- ▶ Imagine something that violates the Landauer form of the law. There is some way to erase information without cost. We turn Bennett's reversible Maxwell's demon loose on a sample of gas. It drives the gas entropy down while filling up its memory with useless data. Now the demon exploits our hypothetical free-erasing process to empty its memory; it returns to its initial state. The only net result is that the entropy of the gas has decreased—a clear violation of the Clausius form of the second law.
- ▶ Or suppose we had an engine that violated the Clausius version, decreasing entropy. It might do this in one of several different ways.
 - For example, imagine that the engine could extract heat from a body and turn 100% of it into work. Energy in the form of work can be used as we like; for instance, we can use it to erase some bits of information.
 - The waste heat from that process would be returned to the body from which the heat was originally extracted. The net result would be that the body had returned to its original state, and nothing else had changed except that we erased some bits.
 - This is a violation of the Landauer form of the second law, which means that they really are the same thing.

Maxwell's Demon in Reality

- ▶ Maxwell's demon is a famous thought experiment, but increasingly, it is becoming a real laboratory experiment. As we gain increasing control over nanoscale systems, we can effectively build devices that act on microscopic information.
- ▶ Consider, for example, an experiment done in 2011 by Mark Raizen's group at the University of Texas. They began with a cloud of ultracold atoms confined in a magnetic trap. This type of atom could exist in two stable internal states, which we may think of as red or green. Across the middle of the volume is a laser beam that would act as a barrier to red-state atoms, but initially, all the atoms are in the green state, so they pass through.
- ▶ Just to the right of the barrier is another laser; let's call it yellow. This is the demon laser. The laser, in a sense, "finds out" where the atoms are and manipulates their states accordingly. When a green atom passes through this beam, it absorbs and emits photons, then turns into a red atom. Therefore, any atom that crosses the middle barrier is soon switched to red and stays on the right side afterward. Eventually, the whole cloud is on the right.
- ▶ Raizen's experiment does not violate the second law; the photons carry away entropy from the color-switching process. But it does provide a way of compressing the cloud of atoms without heating it up, and that's new. It may have applications in everything from medical imaging to isotope separation.
- ▶ We haven't so much "exorcised" the demon from thermodynamics as made a home for it. Maxwell's demon has become our teacher and our guide. It suggests a way to think about the second law more deeply: not simply as a rule about energy transformations but as a profound principle of information.

TERM

eidostate: A neologism (based on the Greek word meaning “to see”) for the state of a thermodynamic system, including all of the information available to Maxwell’s demon, whether it is macroscopic or microscopic.

READINGS

Seife, *Decoding the Universe*, chapters 2–3.

Siegfried, *The Bit and the Pendulum*, chapter 3.

QUESTIONS

- 1 Suppose it was terribly important to build a computer with very little heat dissipation—as few watts as possible. Without actually inventing new technology, what techniques might you try?
- 2 Moore’s law (see Lecture 1) charts an exponential increase over time in computer density (transistors/device), speed, and so forth. To what extent does this depend on a corresponding exponential decrease in heat dissipation?
- 3 Explain why a reversible logic gate must have as many outputs as inputs.

John Kelly, like so many pioneers of the science of information, was an engineer at Bell Labs in the 1950s, around the same time that Claude Shannon worked there. Kelly, however, taught us to look at information theory in a new and entirely different way. For Shannon, information theory was about codes and messages. For Kelly, it was all about gambling. The relation he found between Shannon's theory and making bets opens one of the most controversial chapters in the science of information. As we will see, the same concepts developed to describe communication also have a financial side, with applications from Las Vegas to Wall Street.

Understanding Odds

- ▶ Let's start with a binary horse race. Two horses, X and Y , are at the starting gate. We want to bet on the outcome of the race, but first, we need to know the odds.
- ▶ The idea of odds can be confusing. Odds are used for different things and expressed in different ways. What are the odds for horse X to win? That question might mean two different things.
 - It might be asking the likelihood that horse X will win, in which case, the odds are *statistical* or *probability* odds.
 - The second meaning of *odds* relates to how much money a bet on horse X would pay out if X wins. These are *betting* odds, which are quoted in a couple of different ways.
 - Using the British system, we say that horse X has 4 to 1 odds. This means that on a \$1 bet, if X wins, we are paid \$4, but if X loses we must pay \$1.

- A somewhat more modern system uses **decimal odds**. The same odds on horse X would be quoted as odds of 5.0. The idea here is that we first buy a bet for \$1. If we win, we are paid \$5, which we can think of as our original \$1 plus \$4. For the decimal odds for horse X , we'll use the designation $o(X)$. If we make a \$1 dollar bet on horse X and we win, then we get paid $o(X)$ dollars.
- ▶ Of course, betting odds do have a connection to probability. If $o(X) = 100$, the bookmaker must estimate that horse X is unlikely to win; it's a long shot. If the odds are short— $o(X)$ is not much more than 1—then the bookmaker must think that horse X is very likely to be the winner. The bookmaker's odds are an expression of his probabilities, based on his information.
- ▶ There is an ideal mathematical relationship here. If $q(X)$ is the bookmaker's probability that horse X wins, the ideal decimal odds for X is just the reciprocal of $q(X)$: $1/q(X)$.
- ▶ For our purposes, we will assume ideal odds, and to keep things simple, we will suppose that the odds are even in our binary horse race. The bookmaker thinks that each horse is equally likely to win; $q(X)$ and $q(Y)$ are both $1/2$, and each has decimal odds of 2.0. A \$1 bet on either horse, if successful, yields \$2.
- ▶ Let's imagine that we received a tip causing us to believe that horse X is more likely to win. How much information is actually in a tip? We can answer that question based on Shannon's theory.
 - For the bookmaker, who didn't receive the tip, each of the two horses is equally likely to win. His entropy—the information he lacks about the race—is 1 bit. That's his average surprise about the winner.
 - But the tip causes us to modify our probabilities. We now assign $p(X)$ to horse X and $p(Y)$ to horse Y . If the tip actually tells us for sure which horse will win, then it carries a 1 bit of information. But

the tip might not be quite so informative. After we receive the tip, the information we lack is the entropy, H , the average surprise based on our new probabilities:

$$H = p(X) \log_2 \left(\frac{1}{p(X)} \right) + p(Y) \log_2 \left(\frac{1}{p(Y)} \right).$$

- Initially, like the bookmaker, we lack 1 bit of information. After the tip, we lack H bits of information. Therefore, the tip has conveyed $1 - H$ bits of information.
- In the language of Lecture 7, the tip is like a noisy communication channel. Our residual uncertainty about the race is the noise. If R is the race result and T is the tip: $H(R)$ is the original entropy, 1 bit. $H(R|T)$ is the conditional entropy of the result given the tip; that's H . $I(R:T)$, or just I , is the mutual information, the amount of information the tip gives us about the race results. That equals the difference between the two, or $1 - H$.

► Before we move on, note this fact: From the bookmaker's probabilities, $q(X)$ and $q(Y)$, we can calculate the bookmaker's surprise for each of the horses.

- Surprise is \log_2 of the reciprocal of the probability, or $\log_2(1/q(X))$ and $\log_2(1/q(Y))$.
- We know more than the bookmaker; we have the right probabilities $p(X)$ and $p(Y)$. Thus, the bookmaker is more surprised on average than we are. Mathematically, H is less than or equal to the sum of the p 's multiplied by the logs of the reciprocals of the q 's:

$$H \leq p(X) \log_2 \left(\frac{1}{q(X)} \right) + p(Y) \log_2 \left(\frac{1}{q(Y)} \right).$$

- We call this fact the *information inequality*. The inequality becomes an equation only when the q 's are exactly the same as the p 's.

Exponential Growth

- ▶ John Kelly imagined a series of identical races, one after the other. In each, the bookmaker's decimal odds are 2.0 on each horse. For each race, we get a tip that leads us to adjust our probabilities. Without the tips, we would not expect to win any money in the long run, but with the tips, we can hope to win more often than we lose and come out ahead.
- ▶ Kelly asked two fundamental questions: (1) What is the best betting strategy in the long run? (2) What payoff can we expect, in the long run, from that strategy?
- ▶ We could make a series of \$1 bets that would pay a reasonably steady income, with some ups and downs. Roughly speaking, our wealth would increase linearly, at a constant rate. But once we have accumulated some wealth, those \$1 bets don't make sense. Why play for such small stakes? We should increase our bets and, therefore, increase our returns as our wealth grows. If we do that, the long-term growth of our wealth should be exponential.
- ▶ Kelly sought to harness the force of compound interest; let's consider the mathematics here.
 - We start by investing a certain amount of money, A . Over a period of time, say, a year, the value of the investment increases by a growth factor, G ; A becomes $G \times A$. We often express that as a percentage, such as 25% growth; that means $G = 1.25$. A growth factor of less than 1 means we lose money, and that is, of course, quite possible.
 - After another year, we multiply by another factor of G . Now we have $G^2 \times A$. Year by year, our money grows by a factor of G :

$$\begin{aligned} A &\rightarrow GA \rightarrow G^2A \rightarrow G^3A \rightarrow G^4A \rightarrow \dots \\ \$100 &\rightarrow \$125 \rightarrow \$156 \rightarrow \$195 \rightarrow \$244 \rightarrow \dots \end{aligned}$$

- The key point about compound interest is that the growth factors multiply together. The overall factor for four years is G^4 . The number of years that have passed appears in the exponent of G . That's why mathematicians call this *exponential growth*.

Maximizing Investment

- With this understanding, we can answer such questions as: Which would be better in the long run, an investment of \$10 that grows at 10% per year or an investment of \$1 that grows at 20% per year? The growth factors in the two cases are 1.10 and 1.20. We can use those to calculate the overall growth factors for years into the future:

Year	$(1.10)^n$	$(1.20)^n$
1	1.10	1.20
2	1.21	1.44
5	1.61	2.49
10	2.59	6.19
20	6.73	38.34
30	17.45	237.38

- After 30 years, the 20% growth factor is more than 10 times the other factor; thus, that investment is now worth more in absolute terms, even though it started out much smaller.
- In the long run, the larger growth rate always wins, no matter how things start out. But that is a statement about the long run. Practically speaking, 30 years might be too long to wait!
- Kelly's repeated horse race is similar. He wants to maximize the growth factor of his race investment. But the growth factor varies because the same horses don't always win the race. That's like an investment in

which we get G_1 the first year, G_2 the second year, and so on, as shown below. The G 's aren't all the same; some of them are probably less than 1.0.

$$A \rightarrow G_1 A \rightarrow G_2 G_1 A \rightarrow G_3 G_2 G_1 A \rightarrow \dots$$

- ▶ Let's consider an easy example. In year 1, our growth rate is -50%. The growth factor G_1 is 0.5. In year 2, our growth rate is 100%. The growth factor G_2 is 2.00. Overall, then, we broke even: $G_2 \times G_1 = 1.0$. What is the equivalent annual growth rate? What steady growth factor G for two years would yield the same net return? The answer is obvious, but it's not the average of the G 's.
 - If we average 0.5 and 2.0, we get 1.25, which is too large. What we really need to calculate is the **geometric mean**. For two numbers, that's the square root of their product: $G = (G_2 G_1)^{1/2}$. In our example, that formula gives 1.0, which is the correct annual growth factor.
 - If we do this for a series of n years—or horse races—then the geometric mean is the product of all the G 's raised to the $1/n$ power: $G = (G_n \dots G_2 G_1)^{1/n}$. That's the steady growth rate that gives the same overall return.
- ▶ The same equation looks quite different if we take its logarithm. But first, remember two logarithm facts:
 - If we multiply numbers together, their logarithms add up: $\log_2(xy) = \log_2(x) + \log_2(y)$. The product of growth factors turns into the sum of their logarithms.
 - If we raise a number to a power, that just multiplies the logarithm by that power: $\log_2(x^p) = p \log_2(x)$.
 - Putting it all together, we get a formula for the logarithm of the equivalent steady growth rate: $\log_2(G) = (1/n)[\log_2(G_n) + \dots + \log_2(G_2) + \log_2(G_1)]$.

- The log of G is the ordinary average of the logs of the growth rates. That's the meaning of the geometric mean, written in logarithm language.

Implementing a Betting Strategy

- ▶ According to Kelly, we should choose our betting strategy to maximize $\log_2(G)$. That will give us the best long-term growth rate for our money. How do we do it?
- ▶ Even if we think that horse X is more likely to win, we must not bet all our money on X because X might lose. In the long run, X is sure to lose eventually. When that happens, the growth factor for that race is 0. Thus, the overall growth factor, the product of the factors for all the races, is also 0. That's called the *gambler's ruin*, and we need to avoid it.
- ▶ One way to avoid the gambler's ruin is not to bet some money; another way is to hedge our bets—betting some money on each horse. Because one of them is sure to win, we can never get completely wiped out. It turns out that since our bookmaker is offering ideal odds, the two ways are exactly equivalent. There is no real difference. We will assume that we will always bet all of our money, just dividing it somehow between the horses.
- ▶ We bet a fraction, $b(X)$, of our wealth on horse X and a fraction, $b(Y)$, on horse Y . The two fractions add up to 1, like a probability. Our job is to figure out how to choose $b(X)$ and $b(Y)$ to make $\log_2(G)$ the greatest.
- ▶ Suppose horse X wins. We bet $b(X)$ on that horse and won back \$2 for every \$1 we bet. The money we bet on horse Y is lost. Therefore, our growth factor when X wins is $2b(X)$. When Y wins, the growth factor is $2b(Y)$.

- After many horse races— n of them—the two horses have each won some. X has won about $n \times p(X)$ races, and Y has won about $n \times p(Y)$ races. Now, remember our equation for $\log_2(G)$: $\log_2(G) = (1/n) [\log_2(G_n) + \dots + \log_2(G_2) + \log_2(G_1)]$.
 - In that sum, a certain number of the terms— $n \times p(X)$ of them—will be races that X wins. The rest will be races that Y wins. That simplifies things because n , the number of races, divides out:

$$\begin{aligned}\log_2(G) &= (1/n)[np(X)\log_2(2b(X)) + np(Y)\log_2(2b(Y))] \\ \log_2(G) &= 1 + p(X)\log_2(b(X)) + p(Y)\log_2(b(Y))\end{aligned}$$

- Also, we can separate each logarithm into the sum of two logarithms: $\log_2(2) = 1$, and $p(X) + p(Y) = 1$. Therefore, $\log_2(G)$ is just 1 plus the sum of the p 's multiplied by the logs of the b 's. We can rewrite this in a clever way, remembering that $\log_2(b)$ is just $-\log_2(1/b)$:

$$\log_2(G) = 1 - \left(p(X)\log_2\left(\frac{1}{b(X)}\right) + p(Y)\log_2\left(\frac{1}{b(Y)}\right) \right).$$

- The part in parentheses that we subtracted off is like an average surprise, but it is the average surprise of someone who believes that our betting fractions, $b(X)$ and $b(Y)$, are actually the probabilities for the horses to win. Their average surprise is always greater than H —the entropy of the actual probabilities. That is the information inequality. We want to make the subtracted part as small as possible to make $\log_2(G)$ large. And that average surprise could be made as small as H if the b 's equal the p 's.
- We now have the answer to Kelly's two questions.
 - First, how should we bet? To make $\log_2(G)$ as large as possible, we should choose betting fractions b equal to the probabilities p for each horse. Kelly tells us that if horse X has a 75% chance to win, we should bet 75% of our money on X and hedge our bet by putting the other 25% on Y .
 - Second, what payoff can we expect? That's determined by G , the growth factor for our wealth, per horse race. When we choose the

best strategy, the log of the growth factor is just $1 - H$. But that is exactly the amount of information in the race tip! We called that I , the mutual information: $\log_2(G) = I$, or in exponential form, $G = 2^I$.

Testing Kelly's Relation

- ▶ Kelly's relation between $\log_2(G)$ and I is astounding. It gives us an entirely new way to think about information. At the beginning of this course, we defined *information* as the ability to distinguish between possible alternatives. For Kelly, information is the advantage we have in betting on possible alternatives. Every additional bit of information doubles our long-term advantage. Let's see how it works in a couple of easy cases.
- ▶ Suppose the tip gives us no information. In that case, $I = 0$. After the tip, we still have an entropy $H = 1$ bit. We are as uncertain about the winner as we were before. The best we can do is to divide our bets equally between the horses. Because we always win one of the bets, we get our money back, but no more than that. The growth factor $G = 1$, so $\log_2(G) = 0$, just as Kelly's formula tells us.
- ▶ If the tip tells us everything about which horse will win, $I = 1$ bit, and $H = 0$. We should, therefore, bet everything on the sure winner. We will double our money each time: $G = 2$, which means $\log_2(G) = 1$. That also agrees with Kelly's formula.
- ▶ Now suppose that our tip tells us only a little. After we get it, we estimate that one horse has a 75% chance of winning and the other, 25%. That's how we'll allocate our bets. The new probabilities yield an entropy $H = 0.81$ bits, which means that the tip gave us only 0.19 bits of side information—that doesn't sound like much. According to Kelly, we can make this equal to $\log_2(G)$, so the gain factor $G = 2^{0.19}$, or 1.14. In the long run, we expect to increase our wealth about 14% per horse race—even a pretty poor tip can help!

Alternative Betting Scenarios

- ▶ We've explained Kelly's discovery using a simple example, but he analyzed other situations, as well. For instance, suppose the bookmaker's odds are different. If the bookmaker's probabilities favor horse X over horse Y , then horse X will be given lower odds, even in the ideal case. How should we take these new betting odds into account in placing our bets? Kelly's answer is startling: We should ignore the odds!
 - The fraction of our wealth that we should bet on X is $p(X)$ —our probability—whatever the odds.
 - We always bet according to our information about the result. The change in the bookmaker's odds does affect the payoff, but Kelly's relation between information and growth remains. Every additional bit of information we obtain doubles our long-term gain.
- ▶ Kelly also considered the situation where the bookmaker does not offer ideal odds. He reduces them a little to give himself an edge. Our best strategy is now more complicated. It might be that some part of our money should be kept in cash, rather than wagered. Also, we may not wish to bet on every horse. If the odds are bad enough, it may even be better not to bet at all. In any case, Kelly's information-theoretic analysis tells us how we should bet to maximize $\log_2(G)$, and it tells us how much each bit of side information is worth.
- ▶ Kelly's idea of maximizing $\log_2(G)$ is called a *log-optimal strategy*. It's the one that ensures the greatest growth in the long run, and it's the one that is connected to Shannon's information. If you follow a Kelly log-optimal strategy, you will never go broke, because you never bet everything on any horse that can lose—but you may be in for a wild ride.
- ▶ Not everyone has the nerve to play a log-optimal strategy consistently. Many gamblers use a “half-Kelly” approach instead, which bets only half as much as Kelly advises. In our horse race example, you'd keep half your wealth in cash each time and bet the rest. The gain factor is less, but the ups and downs are much reduced.

- Furthermore, the whole strategy depends on having some side information, that is, on knowing something that the bookmaker does not. We've assumed that the long-term statistics are more like our predictions than his! In much of gambling, the betting odds are set by a parimutuel system. The odds on a horse are set by the total amount of money that people are willing to bet on it. Parimutuel odds represent a "pooling" of information from many people, including some who are very well informed. That makes parimutuel odds hard to beat.

Critiquing the Log-Optimal Strategy

- Claude Shannon later applied Kelly's ideas to the stock market. He sought to devise a log-optimal investment strategy using the mathematical tools of information theory. Of course, the stock market is not quite like a horse race. In the stock market, stocks go up and down in value, usually by a few percentage points. In a horse race, one horse pays off big, but bets on the others are dead losses. Nevertheless, many of the basic ideas carry over.
- One critic of the log-optimal Kelly strategy was Paul Samuelson of MIT, probably the most accomplished and influential mathematical economist of his day. Samuelson did not disagree with Kelly's mathematics; he just disagreed that the math expressed a realistic view of the relationship between people and money.
- First, Kelly's strategy is only optimal in the long run, but that might be very long indeed. For any given finite amount of time, the log-optimal strategy exposes the investor to the possibility of very large losses. To maximize the gain factor, the strategy accepts a great deal of risk.
- The point about risk brings up an even more basic point. According to Samuelson, the log-optimal strategy ignored the way that people actually value money. If you have \$100, then \$10 extra seems like a lot. But if you have \$100,000, that same \$10 is not such a big deal. Thus, the relation between money and value can be complicated. What we really

seek to increase in our lives is value—the things that we value, following our own idiosyncratic ideas.

- ▶ According to Samuelson, the problem with theories based on “information of the Shannon type” is that they make an implicit assumption about the way money is to be valued. That assumption, however, does not match the way that people actually behave. Most people are very willing to trade future money—long-term gain—for security. They are happy to avoid large financial risks. And we can’t say that people are wrong to avoid those risks. After all, money has no value, except the value that people place on it.
- ▶ Samuelson’s critique was, in fact, fairly devastating, particularly among economists. Not many of them take the log-optimal strategy very seriously, but the Kelly idea has never disappeared. It is still a popular theory among information theorists and mathematicians and is yet another striking example of the scope of Claude Shannon’s theory, the basic principles of information.

TERMS

decimal odds: A way of stating the betting odds of a horserace or other gambling game. If a particular horse has decimal odds of 5.0, this means that a bet that costs \$1 will pay winnings of \$5 if the horse wins. (The same odds are sometimes called “4 to 1” odds, using the old British system.) In perfectly fair, ideal betting odds, the horse would have a 1/5 probability of winning.

geometric mean: The square root of the product of two numbers or, for N numbers, the N^{th} root of their product. The average growth rate of an investment or debt is just the geometric mean of the annual growth rates over its lifetime.

READINGS

Kelly, "A New Interpretation of Information Rate."

Poundstone, *Fortune's Formula*.

QUESTIONS

- 1 In our even-money, ideal-odds horserace, we get a tip that tells us that one horse has a 90% chance of winning. How much information is in such a tip? How should we hedge our bets? What is the gain factor for this situation? About how many races would we need to bet on to increase our wealth 1000 times? How many times would we expect to lose in that series of races? How much of our wealth would we lose each time we lost?
- 2 Consider the competing arguments of Kelly (with Shannon) and Samuelson on the advisability of log-optimal strategies. What points seem most important to you?

Turing Machines and Algorithmic Information

In the 1960s, an entirely new theory of information was discovered independently by the most famous mathematician in the world and by a teenaged undergraduate at the City College of New York. The famous mathematician was the Russian Andrey Nikolaevich Kolmogorov, and the undergraduate was Gregory Chaitin, who would later become an eminent computer scientist and researcher at IBM. While Shannon had based his theory on communication—in which a message is encoded and sent to a receiver—Kolmogorov and Chaitin based their new theory on computation—in which a program is executed and produces some output. The theory came to be called *algorithmic information theory*.

The Turing Machine

- ▶ We can imagine all kinds of machines for doing information processing. Even one of Shannon's Boolean logic gates is an extremely simple computer, using bits of input to compute bits of output. But not all computers are equally powerful. One logic gate is too primitive to add numbers together, although a network of logic gates can. That network constitutes a more powerful computer.
- ▶ The limit of computation—the line between what is computable and what is not—seems to depend on which particular computing machine you use. The mathematician Alan Turing, however, showed something quite surprising. Simple computers are all different; they can do different things. But once the complexity of the machine crosses a certain threshold, all computers are the same.
- ▶ Turing's argument is based on simulation. One computer can simulate another. You can run a program on your laptop that is a simulation of Charles Babbage's analytical engine, which means that any computation possible for the analytical engine is also possible for your laptop.

- ▶ Turing showed that the reverse is also true: It is possible to design a program for the analytical engine that simulates your laptop. It might be a huge program with billions of punch cards, and it might take billions of years to run, but every operation in the processor of your laptop can be mimicked by the analytical engine. Thus, any task that your laptop can do can also be done, in principle, by gears and wheels.
- ▶ According to Turing, all sufficiently sophisticated computers are equivalent. They all embody the same universal idea of computation. Because all kinds of information are essentially the same, Shannon could use the bit as a simple common basis for describing information. In the same way, because there is only one universal idea of computation, we can describe computation using one type of ideal computing machine. Turing described such a machine—the simplest computer that is equivalent to all others—and it's now known as a **Turing machine**.
- ▶ A Turing machine can be described as follows: There is a long tape on which symbols can be written. Turing's original design had an alphabet of seven different symbols, but we can use just two—that is, the tape could be a sequence of bits. This single tape is used for all the information storage of the machine: program, input data, scratch space for calculations, final output, and so on. The tape is as long as we need. In addition, there is a very simple device that moves along the tape. This device can go left or right, can read bits from the tape, and can write bits on the tape; it has only 4 bits of internal memory.
- ▶ The machine operates in a series of steps. At each step, the machine reads the bit on the tape and consults its own memory. Depending on what it finds, it writes a 0 or 1 to the tape, moves left or right, and changes its own memory bits. If the internal memory bits are ever all 0, the machine halts. The computation is finished.
- ▶ It might seem that such a machine would be too trivial to do anything interesting. In fact, it is computationally universal. A universal Turing machine can simulate any other computer.

Assumptions about a Universal Computer

- ▶ From this point on, we will assume that we've agreed on a particular universal computing machine. It might be a Turing machine, or it might be some other kind of computer. We will designate it UM for "universal machine."
- ▶ For convenience, we will allow our UM a few extra features. These do not change anything that the UM can do, but they make it easier to discuss how it operates. We will assume that the input to the UM, the program and any initial data, is provided on a separate tape that the machine reads from left to right. The output of the UM will be printed on its own tape. Everything else occurs "inside" the machine, on an internal work tape.
- ▶ Finally, when we discuss a program for the UM, we won't try to describe it by 0s and 1s. Instead, we'll give a heuristic, high-level description that we can understand. Computer programmers call this *pseudocode*—not an actual program but a careful outline from which a program could be written.

Decoding as Computation

- ▶ In a communication system, a codeword is a binary string that allows the receiver to figure out the message. It is a description of the message, written in the code. Decoding is a kind of computation. The receiver takes the codeword as input and computes the message as output.
- ▶ Suppose that our message is itself a binary string, like the one shown below. Actually, s_1 is 1 million bits long, but the rest are just like this string. Our job is to describe the string as briefly as possible.

$$s_1 = 010101010101010101010101010101\dots$$

- ▶ If the receiver is a very simple machine, then we'll have to send the bits one by one, but that seems very inefficient. Intuitively, there is just not that much information in the string. If the receiver is a computer like the

UM, we could describe the string by sending a program, an algorithm for making the string. The UM runs the program, produces the string as output, then halts. The message is decoded. The program might look something like this:

1. Print "01" half a million times.
2. Halt.

Note that this takes advantage of the pattern in the string. The result is a program that is much less than 1 million bits long.

- This is the fundamental idea of algorithmic information theory. The description of a binary string is a program for a UM that prints the string. The information content of the string is measured by the size of the program. A string like s_1 that has a very short program has low information content. Some other string without a short description has a high information content.
 - For example, suppose the million-bit string looks like this:

$$s_2 = 100011100001010011111001\dots$$

- Because there is no pattern in s_2 , there is no short description of s_2 . The shortest program we could send would probably look like the one shown below. This program contains the string itself; it's about 1 million bits long.

1. Print "100011100"
2. Halt.

- Sometimes a pattern exists, but it is hidden. Consider this string:

$$s_3 = 110010010001111110110101\dots$$

- Any statistical test we apply to s_3 will tell us that it is random. The 0s and 1s appear with equal frequency. Each bit is uncorrelated with the last one. We might conclude that s_3 has no brief description and, thus, a high algorithmic information content.

- But in fact, s_3 is the binary expansion of π , the ratio of the circumference to the diameter of a circle. These bits can be calculated by an elegant formula—a short program:

$$\pi = 4 \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots \right).$$

Thus, s_3 has a relatively low algorithmic information content.

Algorithmic Entropy

- ▶ We have our UM, whose programs and output are binary strings. If the program p produces the output s , then halts, we can symbolize that with an arrow: $p \rightarrow s$. Not every program produces an output because not every program actually halts; some get stuck in an endless loop. But if $p \rightarrow s$, then p is a “description” for s as far as our UM is concerned.
- ▶ Any binary string s (program or output) has a length $L(s)$, the number of bits in s . For example, for our binary string s_1 , $L(s_1)$ was 1 million. In contrast, the simple program that generated it— p_1 —was shorter; $L(p_1)$ was much less than 1 million.
- ▶ For any string, there are many different programs that produce it as output. Thus, for string s , we’ll let $P(s)$ stand for the set of all programs p for which $p \rightarrow s$. Some programs in $P(s)$ may be quite long and inefficient; however, there must be a program in $P(s)$ with the fewest bits. That program is the shortest description of s . We’ll call it s^* .
- ▶ The length of s^* is a way to measure the algorithmic information content of the string s . This is often called the *Kolmogorov complexity*, but we’ll use the term **algorithmic entropy**. We’ll designate it with K . Its definition is simple: $K(s) = L(s^*)$. Algorithmic entropy is the length of the shortest description—the shortest program that prints the right output, then halts.

- ▶ This new entropy plays the same role in the new information theory that Shannon's entropy played in the old, but notice how different the two are. Shannon's entropy does not make sense for a particular string of bits. Entropy is a property of an information source. There are many possible messages, each with its own probability. Entropy measures the size of that universe of possibilities. In contrast, the algorithmic entropy makes sense for any particular string of bits. The strings themselves can have a higher or lower information content, according to whether they require longer or shorter descriptions.
- ▶ The two entropies are related to each other. For a source that produces binary sequences, the Shannon entropy is approximately the average of the algorithmic entropy, taking an average over all the possible sequences that the source might produce: $H \approx \text{ave}(K)$. Shannon entropy is a way to estimate the algorithmic entropy, on average.

Defining *Randomness*

- ▶ The algorithmic entropy K gives us a completely new way to define *randomness*. The old idea of randomness had to do with probability. But probability, like Shannon's information theory, is really not about particular events. A "random" variable X imagines a universe of possible events, not one in particular.
- ▶ In algorithmic information theory, we can consider a particular string and say whether it is random or has a pattern. If string s has a pattern—even a subtle one, like the bits of π —then it can be described by a program much shorter than s itself. $K(s)$, the length of the shortest program s^* , is much less than the length of s : $K(s) = L(s)$.
- ▶ On the other hand, a string is random if there is no short way to describe it. Of course, you can always describe a binary string just by listing it: the program that says "Print s , then halt." That program has about the same length as s itself. Therefore, s is random if there is no shorter way than that to describe it. $K(s)$, in other words, is about equal to the length of s : $K(s) \approx L(s)$.

- ▶ This definition of randomness has nothing to do with probability. Indeed, Kolmogorov believed that the idea of information was more fundamental than the idea of probability.

Algorithmic Information and Probability

- ▶ There is an amusing way of connecting algorithmic information to probability. Recall the old idea of having a monkey bang away at a typewriter. It hits the keys randomly—with equal probability. We can then calculate the likelihood (very tiny, of course) that the monkey by chance writes one of Shakespeare’s plays.
- ▶ Let’s play this game a little differently. The monkey will be a programmer for our universal computer. Its keyboard will just have two keys, 0 and 1. It hits them with equal probability, $1/2$. Whatever the monkey types is given to the UM as a program. We run it and see what the computer does.
- ▶ Note that we have assumed the UM reads its input tape—where the program is—from left to right. Thus, if the first 100 bits of the tape form a valid program, none of the rest of the bits will even be read by the computer. If p is a valid program for the UM, then p followed by some additional bits—such as $p + 0110100$ —is really the same program. Only the first $L(p)$ bits of the input tape matter.
- ▶ What is the probability that the monkey programmer will write program p ? Each bit in p has probability $1/2$; thus, the overall probability is $(1/2)^{L(p)}$, or $2^{-L(p)}$. That’s the “monkey probability” of program p .
- ▶ The monkey is sure to type some kind of program, and the computer will do something. Thus, the sum of all the monkey probabilities is 1. We can write that fact using the sigma notation for sums:

$$\sum_p 2^{-L(p)} = 1.$$

- ▶ Why is s^* so dominant? Consider two programs, p and p' , both of which produce s . The program p is 500 bits long, while p' is 510 bits long. That does not seem like much of a difference in length, but it makes a huge difference for monkey probability. That 10-bit difference means that p' is 2^{10} times less likely—about 1000 times less likely.
- ▶ By far the most probable way for the monkey to produce s is for it to type program s^* . This means that it is not all that unlikely for the monkey to produce a string of a million 1s. The shortest program that produces that string is very short. We can just barely imagine a monkey typing that program by accident. The total monkey probability $M(s)$ for string s is, therefore, about $2^{-L(s^*)}$, which is 2^{-K} .
- ▶ We take the logarithm of this equation and solve for the algorithmic entropy $K(s)$:

$$K(s) \approx \log_2 \left(\frac{1}{M(s)} \right).$$

That log of reciprocal probability is what we called “surprise” in the Shannon theory. Therefore, it makes perfect sense to say: Algorithmic entropy equals the monkey surprise.

Artificial Intelligence

- ▶ This is much more serious than it sounds. In fact, it can shed new light on a centuries-old philosophical principle, **Ockham’s razor**, which states that when we are faced with a choice of hypotheses, we should choose the simplest one. The simpler theory is not necessarily the true one, but it is more likely to be true.
- ▶ In thinking about artificial intelligence, Ray Solomonoff wanted to teach computers to reason as we do: inductively, starting with data and choosing hypotheses to explain the data. Solomonoff said that the data can be represented by a string of bits, s .

- A theory is essentially a program p that describes the data s . That theory could be—and should be—much more compact than the original data.
- If there is a great deal of regularity and pattern in the data—if the sun comes up every day and apples always fall to the ground when you drop them—that will be reflected in the fact that the description p can be made much, much shorter than s .
- ▶ Any theory p is a program within $P(s)$. Ockham's razor advises that the computer should choose the shortest one, which is s^* . As we saw, that one is more likely in monkey probability! If a monkey were to generate theories at random, and it happened to produce a theory that matched the data s , then it would be most likely to get there by typing s^* . That's the sense in which s^* is the most likely theory.
- ▶ Of course, if there are two or more theories that have the exactly same number of bits, they will be equally likely in Solomonoff's monkey version of Ockham's razor. The computer will have no basis to choose between them. But as we saw, even a few bits one way or the other can make a significant difference in probability. The monkey odds are strongly in favor of the shortest description, s^* .

Applying Algorithmic Information

- ▶ Algorithmic information theory helps us understand Maxwell's demon. As we discussed, Maxwell's demon acquires information about the microstate of a physical system. It uses that information to guide the system—opening and closing a trapdoor between gas containers, for example—so that the thermodynamic entropy of the system decreases.
- ▶ But Bennett showed that the second law of thermodynamics is safe. At the end of the day, the demon must erase its stored memory record. The thermodynamic cost of that erasure, given by Landauer's principle, is enough to prevent any net decrease in entropy.

- ▶ Yet perhaps that explanation leaves us a bit uneasy. The second law is safe in the end, but what about in the middle? The demon has not yet dissipated any waste heat. Its memory is full of the microscopic information it observed. The entropy of the gas is reduced. Isn't that at least a temporary reduction in entropy?
- ▶ That's the question that caught the interest of Wojciech Zurek of Los Alamos National Laboratory. He reasoned as follows: When the demon acquires its information, it reduces entropy elsewhere. When the demon erases its information, it increases entropy elsewhere. The demon's information must be a form of entropy! Yet the memory record is some particular string of bits. It is not "missing microstate information," like the ordinary entropy. It's not missing at all. The demon knows it. It's part of what the demon sees, part of its eidostate. How can that have entropy?
- ▶ According to Zurek, it has algorithmic entropy. The total thermodynamic entropy must be the sum of two parts. The first part is k_2 (the Boltzmann constant) multiplied by $\log_2(M)$ (the number of possible microstates). That's k_2 multiplied by the Shannon entropy of the microstate, given all available information. But there is also a second part to the entropy. That is k_2 multiplied by $K(m)$, the algorithmic entropy of the demon's memory record. Here is Zurek's formula: $S = k_2 \log_2(M) + k_2 K(m)$. The total physical entropy is the sum of Shannon entropy and algorithmic entropy.
- ▶ When the demon's memory is empty—all 0s— $K(m)$ is tiny. When the demon acquires its microscopic information about the gas, the algorithmic part of the entropy increases enormously.
- ▶ Zurek showed that the total entropy, including the algorithmic part, never goes down on average at any stage. As the demon acquires and uses information, ordinary entropy decreases, but K increases. As the demon erases its data, K decreases, but ordinary entropy increases. The second law holds true throughout.

- Zurek's formula is an update to Boltzmann's original formula, $S = k_2 \log_2(M)$. It takes into account the information that we possess, as well as the information we lack. We've never had to think of this before now, because in ordinary circumstances, the information we possess is negligible. After all, that constant k_2 has a value of 10^{-23} joules/kelvin per bit. Even the memory capacity of the human brain, around 10^{16} bits, makes only a tiny thermodynamic contribution. But Maxwell's demon teaches us something new: The algorithmic entropy K , the Kolmogorov-Chaitin information content of a memory record, has a real meaning for the laws of physics.
- There's only one problem with Zurek's formula: The algorithmic entropy K of the memory record m is uncomputable. Even a universal computer can never do it. We'll take up this story of the inherent limits of computation and logic in the next lecture.

TERMS

algorithmic entropy: The length of the shortest program on a universal computer that prints a specified string as output and then halts. Formulated by Andrey Kolmogorov and Gregory Chaitin as the basis for algorithmic information theory, in which the information content of any object is simply the length of its shortest description on a computer. The algorithmic entropy of a binary string is an uncomputable function, thanks to the **Berry paradox**.

Ockham's razor: The logical principle, formulated by the 14th-century English philosopher William of Ockham, stating that the simplest theory is the one most likely to be true. Centuries after Ockham, Ray Solomonoff reformulated the principle using "monkey probabilities" from algorithmic information theory.

pseudocode: An informal description of a computer program, readable by humans and containing all the essential details of an actual program.

Turing machine: An extremely simple but universal computing machine described by Alan Turing in 1937 as a very small and simple device that moves forward and backward along an unlimited data tape, reading and writing symbols according to a simple rule. Such a machine can be shown to be equivalent to an extremely sophisticated computer in that the two types can simulate each other and, thus, perform exactly the same computations.

READINGS

Gleick, *The Information*, chapter 12.

Hofstadter, *Gödel, Escher, Bach*.

QUESTIONS

- 1 A Turing machine has an infinite tape, that is, an unlimited memory. A laptop computer has only a finite memory. In what sense is it a “universal” computer in Turing’s meaning?
- 2 Suppose we consider a long binary string produced by a Shannon-type information source. Why is it reasonable that the average of $K(s)$ is about H of the source?
- 3 What does the word *random* mean? What does it refer to? How is this related to data compression?

Uncomputable Functions and Incompleteness

LECTURE 20

Shannon's information is based on codes and communication. Algorithmic information is based on descriptions and computation. This new way of thinking about information brings with it new insights on many things: the meaning of randomness, Ockham's razor for choosing between hypotheses, and even Maxwell's demon and the second law of thermodynamics. Yet algorithmic information is beset by a strange impossibility, one that connects this new concept of information to the very foundations of logic and mathematics. We'll examine that impossibility in this lecture.

The Berry Paradox

- ▶ The **Berry paradox** was first posed by an Oxford librarian, G. G. Berry, who lived about 100 years ago. It's a paradox about the positive whole numbers (integers). We can describe integers by phrases in English, such as "four" (4), "one plus one plus one" (3), "the product of all the integers up to 10" ($10! = 3,628,800$), and "the largest prime number less than 1 million" (999,983). Every integer has a description, but large integers usually need many words to describe them.
- ▶ Consider the following description: "the smallest integer than cannot be described in 12 words or less." There certainly are integers that cannot be described so succinctly; thus, it follows that there must be a smallest such integer. Of course, here's the problem: That phrase itself has 12 words. Thus, any integer that is described by that phrase cannot be described by that phrase. That's the Berry paradox.

- ▶ The Berry paradox is a warning. It tells us that the concept of *description* is fraught with unseen pitfalls. Yet in algorithmic information theory, the whole concept of information is based on the idea of description. As we explore that concept of information, we must be very careful indeed. It is ironic that the subject of information, which is at root about what we know, brings us face to face with what we can never know.

Computer Programs and Subprograms

- ▶ We have said that a binary string p counts as a “description” of another string s provided that p is a program on a standard universal computing machine (UM) that prints s as output, then halts. Among all the descriptions of s , there is a shortest one called s^* , and the length of s^* is the algorithmic entropy $K(s)$.
- ▶ This led to a neat definition of randomness: A string s is random if it has no description much shorter than itself. Its entropy is about equal to its own length: $K(s) = L(s^*) \approx L(s)$.
- ▶ It turns out that most strings of a particular length are random. Consider strings that are 100 bits long. There are 2^{100} of them, which is a fairly large number. We’ll say that s is random if $K(s)$ is at least 90, that is, s^* is at least 90 bits long.
- ▶ How many non-random 100-bit strings can there be? Each of these is describable by a program less than 90 bits long, which means that the number of such strings is limited by the number of such programs. That number of programs is less than 2^{90} , which is also a large number, but it’s much smaller than 2^{100} . In fact, their ratio is about 0.001:

$$\frac{2^{90}}{2^{100}} = 2^{-10} = \frac{1}{1024} \approx 0.001.$$

- ▶ At least 99.9% of strings with 100 bits are random by this definition. In other words, almost all strings are algorithmically random. The ones with patterns are the rare exceptions.

- The string s might or might not be random, but its shortest description, s^* , is itself always random. Why?

- Suppose s^* were not random. Then its own shortest description— s^{**} —would be much shorter than s^* . Then we could make a new program, called program r :

1. Run s^{**} .
2. Run the output of step 1 as a program and print its output.
3. Halt.

- Program r uses another program, program s^{**} , as a module, a functional part of the whole. Computer programmers call that a *subprogram*. Also, r generates a subprogram in step 1 that it then runs in step 2. That's not a problem because a program or subprogram is just a binary string.
- Program r is really just s^{**} plus a few extra instructions; thus, the length of program r is about equal to the length of s^{**} , which is, by our assumption, much shorter than s^* .
- But look what program r does. In step 1, it actually generates s^* . Then, in step 2, it runs s^* and produces s . In other words, r is a UM description of the original string s . That means it cannot be shorter than s^* , which is the smallest such description. Thus, our assumption is wrong: s^{**} is not much shorter than s^* ; therefore, s^* has no much shorter description. It is random.
- If that seems surprising to you, think of it this way: The difference between the length of a string $L(s)$ and its algorithmic entropy $K(s)$ can be thought of as a kind of “algorithmic redundancy.” It's a measure of how many extra bits we use to write s rather than s^* , its shortest description. When we go from s down to s^* , we squeeze out all that redundancy, leaving none in s^* . That's what makes it algorithmically random.

- ▶ The analogy to Shannon’s information theory is fairly clear. If we have an information source with some redundancy, we can use a code to represent the output of the source with a smaller number of bits. That’s called *data compression*. Thus, replacing s with its minimal description s^* is an algorithmic version of data compression. In fact, it’s perfect data compression because no binary string shorter than s^* could possibly be a description of s . A program that could do perfect data compression—that could convert any string s into its minimal description s^* —would be a very useful thing. But as we will see, such a program is impossible.

- ▶ When Alan Turing worked out the theory of universal computation, he realized that there are things that even a universal computer can never do: uncomputable functions. A function is a mathematical rule that takes an input and produces an output. We can design programs on our UM that do the same sort of thing.
 - We begin with a partial program f , a binary string containing some computer instructions. But f by itself is not quite a complete program. We have to add some additional bits as the input data for the program.

 - The complete program for the UM is something like (f,s) —the bits of f , followed by the bits of the function input s . When we run that on the UM, it produces an output string, which we call $f(s)$: $(f,s) \rightarrow f(s)$.

- ▶ How do we describe f itself, apart from its input, s ? First, we make a note that there will be an input, to which we give a temporary name, x . Then, we describe what to do with x . For instance, suppose the input is a number, and f computes the square of the number. We’d write the program f as shown below. (This is the human-language version of some binary program f .) When the computer reads f followed by an actual input s , it applies this procedure to s and outputs $f(s)$ —in this example, the square of s .

```

Program  $f$ : (input  $x$ )
1. Calculate  $x^2$  and print it.
2. Halt.

```

- We can create function programs of all kinds. And since the program and its input are both just strings of bits, we can play some games with the idea. For instance, we can give the computer (f,f) so that it will calculate $f(f)$. With this program, a program for determining the size of data files can be instructed to determine its own size.

Will a Given Program Halt?

- Not all computer programs ever come to a stopping point; some get stuck in an infinite loop. And some functions can be like that, too. A function f might calculate output and halt for some inputs but get stuck for others. For computer programmers, it would be handy to know in advance whether or not a given program is one of those that will run forever or will eventually finish. For the UM, we would like to know whether program (f,s) —function f acting on input s —ever halts.
- As a logical fact, every program (f,s) either will or will not halt. The answer is 1 bit of information: yes or no. Thus, for every (f,s) , there is either a 1 or a 0, depending on whether it halts or not. That's the *halting function*, and it certainly does exist mathematically. It's a function we would like to be able to compute.
- As a thought experiment, Alan Turing imagined a program h to calculate the halting function. This program works as follows: It takes two inputs—the program for the function f and its input s , then it analyzes f and s carefully. It outputs 1 if the program (f,s) will eventually halt and 0 if it will run forever. We assume that the program h never gets stuck itself and always gives the right answer.
- The halting function program h is represented by some binary string, just like any other function on the UM. To run it on the computer, we would make a complete program out of the program h , then the program f , and then the input s . That is, we give the computer a complete program (h,f,s) .

- Once we have the program h , we can use it as a subprogram in a larger program. For instance, we could use it to prevent the computer from ever getting stuck in a computation. It would see that coming and avoid it. Here is program g :

Program g : (input f, s)

1. Compute $h(f, s)$.
2. If $h(f, s) = 1$, then calculate (f, s) and print the result.
3. Halt.

- To compute a function f on input s , we run the complete program (g, f, s) . Here's what g does: Before trying to compute $f(s)$, the computer first checks to see if (f, s) halts. If so, it does the calculation and prints the answer. If not, it quits right away. The computer never gets trapped in an infinite loop.
 - Let's consider a new program, called t (for Turing). This takes an input x and does something a little odd with the halting function, as shown below:

Program t : (input x)

1. Compute $h(x, x)$.
2. If $h(x, x) = 0$, halt. If $h(x, x) = 1$, loop forever.

- Program t is a function. It takes as its input the program for some other function f . The computer is actually running the complete program (t, f) .
- This program uses the halting function program h to determine whether f will get stuck if it is given itself as input, that is, whether the program (f, f) halts. If (f, f) does not halt, then t does halt. If (f, f) halts, t keeps running in an endless loop.
- In short, the program (t, f) always does the opposite of what (f, f) would do. If one halts, the other does not and vice versa.

- ▶ Turing then asked: What happens when f is actually t itself? What happens to the program (t,t) ? Remember, because t is a binary string, (t,t) is a perfectly sensible complete program for the computer. But the program is designed so that (t,t) halts if and only if (t,t) does not halt. It's a logical contradiction!
 - This reminds us of the story of the barber who lives in a village. Some men in the village shave themselves; they are self-shaving men. The barber's rule is that he shaves only those men who are non-self-shaving and no others. But who shaves the barber? If he is self-shaving, then by his own rule, he cannot shave himself. But if he is non-self-shaving, he must shave himself.
 - In the same way, some functions f halt when they receive themselves as input. They are self-halting. The others are non-self-halting. Program t is designed to halt for exactly those programs that are non-self-halting. Is program t self-halting or non-self-halting? Neither one is possible.
- ▶ According to Turing, the only place we could have gone wrong is when we assumed that we had a program h to compute the halting function; therefore, no such program can exist. The halting function—which is well-defined mathematically—is not computable by any program.
 - If we tried to write a program to tell whether another program halts or not, no matter how our program worked, it would have to fail sometimes. Either it would make mistakes by giving the wrong answer, or it would sometimes get stuck and be unable to decide.
 - Turing's proof that the halting problem is an uncomputable function—that no universal computer can solve it—is one of the greatest mathematical discoveries of the last century. It would be no exaggeration to say that the whole modern field of computer science has its origin in Turing's work on universal machines.

Algorithmic Entropy as an Uncomputable Function

- ▶ Uncomputable functions are fascinating, but do they say anything about information? In fact, one uncomputable function is the whole basis for algorithmic information theory, the algorithmic entropy K .
- ▶ As Turing did for the halting function, we will assume for the sake of argument that there is a program k that calculates $K(s)$. If we provide our UM with program k followed by a string s —that is, the complete program (k,s) —then it will compute $K(s)$, print that out, and halt.
- ▶ Program k itself has some length, $L(k)$. It might be quite a long program—perhaps 100 million bits long. But however long it is, we could come up with a number (N) that is more than 1 billion times larger than $L(k)$: $N > 10^9 \times L(k)$.
- ▶ Next, we must note that it is relatively easy to generate all binary strings, one by one, starting from the shortest. That is, we start with 0, then 1; then 00, 01, 10, 11; then 000, 001, and so on. First we get all the 1-bit strings, then all the 2-bit strings, then all the 3-bit strings. This procedure is not very complicated, and if we keep it up, eventually, we hit every binary string of every length.
- ▶ Now we can write a new program, which we will call b .

Program b :

1. Generate the next string s .
2. Compute $K(s)$ using k .
3. If $K(s) < N$, go back to step 1. Otherwise, print s and halt.

- ▶ How large is program b ? It is all fairly simple. The only possibly complicated part is the subprogram k , which might be very large. Thus, in rough terms, b is not much longer than k itself: $L(b) \approx L(k)$.

- ▶ What exactly does b do? It starts with the string 0. This has a small value of K , so it tries again. It generates the string 1. This too has small K , so it tries again. It generates the string 00 and so on. Program b runs for a very, very long time. But eventually, it will generate a binary string s whose algorithmic entropy $K(s)$ is at least as large as the huge number N . It then prints s and halts.
- ▶ We have established three things about program b : (1) It is a description of the string s . That is, it (eventually) prints s and halts. (2) The algorithmic entropy of s is at least N : $K(s) \geq N$. (3) The length of b is about $L(k)$, which is much less than N . These cannot all be true! If $K(s)$ is at least N , then the shortest description of s must be that long. But b , which is much shorter, is also a description of s . We have a contradiction.
- ▶ Program b really is a computer version of the Berry paradox. In words, program b is a description for the first binary string that cannot be described in fewer than N bits. But by assumption, b itself has far fewer than N bits.
- ▶ The problem here is that there is no program k that can compute algorithmic entropy. $K(s)$ is an uncomputable function. Thus, our fundamental measure of information—information based on computation—turns out to be impossible to compute!

Implications of Uncomputability

- ▶ The conclusion we have reached is rather strange, and it has some strange consequences. For example, recall Wojciech Zurek's formula for thermodynamic entropy.
 - Zurek considered a system that included Maxwell's demon as a kind of computer. The demon possesses information, stored in its memory. The binary string m represents that memory record. It has an algorithmic entropy, $K(m)$.

- Given the demon's memory record, there are a great many microstates (M of them) consistent with that information—the demon's eidostate. According to ordinary information theory, the missing microstate information is the Shannon entropy, $\log_2(M)$.
 - According to Zurek, total thermodynamic entropy is the Boltzmann constant k_2 multiplied by the sum of the Shannon and algorithmic entropies: $S = k_2 \log_2(M) + k_2 K(m)$.
 - As the demon acquires and disposes of information, each entropy term might increase or decrease, but the total can never decrease on average. That's the second law of thermodynamics, as it applies to Maxwell's demon.
 - But now we know that algorithmic entropy is uncomputable. Therefore, accepting Zurek's formula, the total thermodynamic entropy is uncomputable. Thus, there is an element in the second law of thermodynamics that we can never compute.
- Why does this matter? Suppose Maxwell's demon is at the stage where it wants to erase its memory record, m . Every bit that it erases will cost energy and produce an entropy of k_2 in the environment. That's Landauer's principle.
- Naturally, the demon does not wish to use any more energy than necessary. Thus, before it erases its memory, the demon will do data compression on it, so that it erases a smaller number of bits. This must be a reversible computation, of course. The demon wants to find the smallest bit-string from which the original memory m can be reconstructed. And, of course, that's just m^* , the shortest description of m on a universal computer. That m^* is $K(m)$ bits long.
 - What the demon wants is perfect data compression—a program that turns m into m^* —but no such program can exist. If it did, we could compress a string s to s^* , then just count the bits in s^* . That would compute the algorithmic entropy $K(s)$. But because we know that K is uncomputable, perfect data compression must be computationally impossible.

- The actual data compression program used by the demon will compress m to m' . The length of m' might be as small as m^* , that is, the data compression might be perfect. But sometimes, m' will be longer than m^* . That means the demon will wind up erasing more bits than $K(m)$. That will expel more energy as waste heat and will increase the overall entropy.
- The demon wastes additional energy because certain mathematical functions are uncomputable. Zurek calls this conjectured effect *logical friction*.
- ▶ Uncomputability is also closely connected to a profound fact about the nature of mathematics: Kurt Gödel's famous incompleteness theorem. Gödel proved that any formal mathematical system of axioms and logical rules, provided it is as sophisticated as simple arithmetic, must contain statements that are neither provable nor disprovable from the axioms. In other words, there are undecidable propositions.
 - These are something like the uncomputable functions for a universal computer. Indeed, the connection is more than an analogy.
 - A computer is itself a formal mathematical system, and a formal mathematical system can be programmed on a computer.
- ▶ Gregory Chaitin, one of the founders of algorithmic information theory, has taken this idea quite far. He imagines a computer given a description of some mathematical system, say number theory. All the axioms about numbers and all the logical rules are expressed as a string of bits, called n . Then, the computer is given a proposition p about numbers. Its job is to either prove or disprove p . When p is undecidable, the computer never halts.
 - According to Chaitin, the basic problem is that the axioms in the bit-string n constitute only a finite amount of information. Some propositions actually entail more information than the axioms themselves and, thus, can never be proven from them. For Chaitin, Gödel's famous theorem about mathematics is actually a deep fact about information. This is nowhere more strikingly confirmed than in the almost inconceivable properties of the number that Chaitin calls Ω .

- This number is actually fairly easy to define. Once again, we imagine that our computer is being programmed by a monkey. The monkey types a random string of 0s and 1s. The likelihood that the monkey types a particular program p is $2^{-L(p)}$.
- Some programs that the monkey types will eventually halt. Others will run forever. The number Ω is the total monkey probability that the program halts. In symbols, we just add up all the probabilities for the halting programs:

$$\Omega = \sum_{p \text{ halts}} 2^{-L(p)}.$$

- The number Ω is somewhere between 0 and 1. It has a binary expansion, an infinite series of bits, something like: .00111100000101 However, Ω is an uncomputable number. No program could produce the bits of Ω , one after the other, as output. Indeed, any initial segment of Ω is algorithmically random. The first million bits of Ω , for instance, have no description less than about 1 million bits long.
- We can find an approximation, Ω' , that is smaller than Ω , and with more work, we can push Ω' closer to Ω . Every time a program halts, that tells us one of the monkey probabilities in the Ω sum. If we add up the monkey probabilities for all the halting programs we know so far, we could call that Ω' :

$$\Omega' = \sum_{\substack{p \text{ known} \\ \text{to halt}}} 2^{-L(p)} < \Omega.$$

Given that there are some other halting programs that we haven't identified yet, we know that Ω' is a little less than Ω . The problem is that we never can know exactly how close Ω' is to Ω .

- The number Ω contains an astonishing amount of useful information. For instance, it turns out that knowing the first N bits of Ω would allow us to solve the halting problem for all programs up to N bits long. That's because we could just test all the N -bit programs at once and calculate an estimated Ω' for Ω by keeping track of which programs halted. Eventually, Ω' would match those N bits of Ω . That means Ω' would be within 2^{-N} of the real thing, and that would mean that none of the rest of the N -bit programs would ever halt.
- Once we can solve the halting problem, we can do many more uncomputable things. With N bits of Ω , we can calculate the algorithmic entropy of any string up to N bits long. In fact, we can solve any solvable mathematical problem that can be stated in N bits.
- The monkey halting probability Ω is, in fact, a maximally compressed compendium of all mathematical information. But that compendium is forever beyond our reach. Even our deepest mathematics has, as it were, revealed only the first few dozen bits, and no one can predict what the next ones might be. Chaitin says that this means there can be no final “theory of everything” in mathematics. There is an infinite amount of information yet to be discovered.

TERM

Berry paradox: The paradox contained in the following question: What is the smallest integer that cannot be described in 12 words or less? Given that some such numbers exist, there must be a smaller one. However, this 12-word phrase itself appears to describe that number, so it is describable in 12 words after all. A computational version of this paradox shows that the algorithmic entropy is uncomputable in general.

READING

Chaitin, *Thinking about Gödel and Turing*, “Randomness and Mathematical Proof” and “The Limits of Reason.”

QUESTIONS

- 1 Can you think of another possible example of a function program that might act on itself as input?
- 2 Based on our discussion of algorithmic information and uncomputability, why is the Berry paradox not a logical contradiction?

In the last years of the 19th century, the German physicist Max Planck was concerned with entropy. The entropy formulas of Boltzmann and Gibbs worked beautifully for the thermodynamics of gases, solids, and liquids. But when Planck tried to use the same ideas for radiation—light—his calculations went badly awry. What Planck eventually discovered in 1900 was something quite unexpected. The entropy formulas were correct; what was wrong was our understanding of the microscopic world. Planck's discovery was the beginning of the greatest revolution in the history of physics: the quantum revolution. This revolution is still in process, and right now, some of its most exciting events are taking place in the science of information.

The Beginnings of Quantum Information Theory

- ▶ Earlier, we made a distinction between digital and analog information. Digital information refers to discrete alternatives, such as the 0 and 1 of a binary digit. Analog information is about variables that can have a continuous range of values, such as the voltage in an electrical signal.
- ▶ According to classical physics—the ideas of physics that prevailed before quantum theory—information in the physical world is all analog information. Every physical variable—the position of a particle or the frequency of a wave—is continuous. Digital information is a convenient simplification and sometimes useful, but it's all analog underneath.
- ▶ What Planck found is that the microscopic world is far more digital than we imagined. A light wave does not have a continuous range of energies but only discrete values; it is *quantized*. The indivisible packets of energy are called photons.

- ▶ This has practical implications for communication. Suppose we want to send signals via electromagnetic waves in an environment where there is essentially no noise. Without noise, we should be able to send our message using almost no energy. The amplitudes of the waves will be extremely low, but they will carry the information. What matters is the signal-to-noise ratio.
- ▶ Quantum physics reaches a different conclusion. At such low wave amplitudes, the energy of the signal will not reach the receiver as a smoothly varying function but as a series of discrete photons that arrive randomly. The photon randomness will introduce noise of its own—*shot noise*. And that inescapable quantum noise will be what limits our capacity to communicate.
- ▶ Physicists and engineers learned how to calculate the quantum noise level in specific situations, but it wasn't until the 1970s that we gained a more comprehensive understanding, thanks to the brilliant work of the Russian mathematician Alexander Holevo.
 - Holevo considered this general problem: We prepare some particles in quantum states that encode our message and send them to a receiver. The receiver makes some kind of measurement to extract the information. How many bits can we send in this quantum communication channel?
 - Holevo derived an absolute upper bound on Shannon's mutual information for the channel. We can never exceed Holevo's bound, and sometimes, we can't even reach it. In other words, we might not be able to reach the ceiling, but there is definitely a ceiling.
- ▶ In the early 1990s, scholars at Williams College were working on a research problem that involved staring at Holevo's ceiling. Perhaps, with the right coding and decoding process, it would be possible to get very close to Holevo's bound. Then, Holevo's upper bound would equal the Shannon information capacity of a quantum channel. Proving this would tell us exactly how many bits that quantum channel was worth. Later,

these researchers decided that for a quantum situation, a new quantum idea of information was needed, and that information could be measured in quantum bits—***qubits***.

- ▶ At the same time, many other ideas that researchers were working on made sense as “quantum information,” including **quantum entanglement**, quantum cryptography, and quantum computing. If we want to build a new quantum information theory, we should start by trying to find an equivalent to Shannon’s first fundamental theorem (the data compression theorem): The Shannon entropy equals the number of binary digits needed to carry a message. What was needed was quantum versions of all the pieces of Shannon’s theorem.

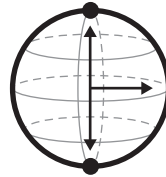
Quantum Equivalents of Shannon’s Terms

- ▶ A bit is any classical system that has two distinct states, which we can label 0 and 1. A qubit is a quantum system with two distinct states. For example, a photon can be horizontally polarized or vertically polarized; thus, we can label the horizontal state 0 and the vertical state 1.
- ▶ A photon polarization is an example of a qubit. Because the qubit is a quantum system, there are also many other qubit states, called *superposition states*, that are strange combinations of 0 and 1. Photon polarization along some other axis is a superposition of horizontal and vertical. And many qubits can be in a collective superposition called an *entangled state*. All of these are built out of the basic states 0 and 1, but there are many more possibilities for a qubit.
- ▶ A quantum message is simply the state of some quantum system. We imagine that different states are produced with different probabilities—that’s our “quantum information source.” The idea is that Alice wants to convey the quantum state she has to Bob by sending him some qubits. If Alice’s initial state is a superposition or an entangled state, all of that must be faithfully carried by the qubits.

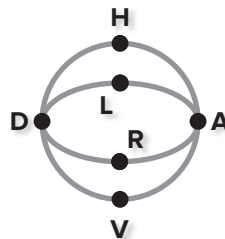
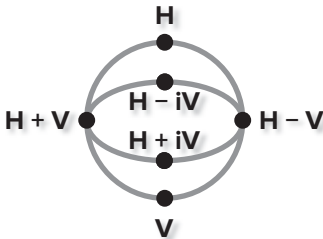
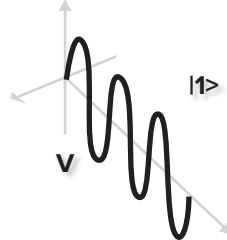
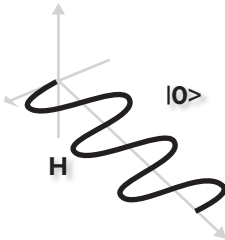
• 1

• 0

Classical Bit



Qubit

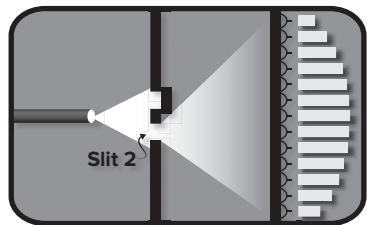
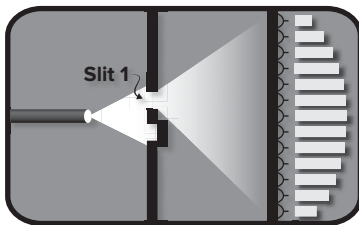


- This brings us to an interesting point. Previously, Bill Wootters and Wojciech Zurek had proved that it is impossible in general to make a perfect duplicate of the quantum state of a system. That was called the **quantum no-cloning theorem**.
- In the new language, we would say that quantum information cannot be copied. Any process that can successfully copy the 0 and 1 states will fail for superpositions. Thus, our coding and decoding processes must be a little different.

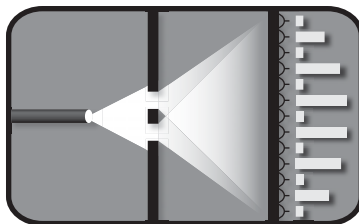
- In Shannon's theory, when Alice sends Bob a message, she can retain a copy for herself. But if she sends a quantum message, that process cannot leave any copy behind. The information is transferred, not shared.
- ▶ Finally, the Shannon entropy is not the right measure of quantum information.
 - Suppose we have a photon with two possible polarizations: vertical and 10 degrees away from vertical. Because they are equally likely, the Shannon entropy is 1 bit.
 - But those two states are so close to each other that they cannot be reliably distinguished by any measurement. They are not as physically distinct as 0 and 1. This means that the Shannon entropy is far too high.
 - Luckily, John von Neumann had already figured out the quantum version of the Gibbs formula for thermodynamic entropy, and we can adapt it as an information entropy. Von Neumann's formula gives just 0.006 bits for our photon polarization example.
- ▶ Putting these pieces together, here is how the quantum version of Shannon's first fundamental theorem ought to read: The von Neumann entropy equals the number of qubits needed to transfer a quantum message. That is the quantum data compression theorem.
- ▶ If you pick up a textbook on quantum information, you will find a mixture of the familiar and the strange. Some of the concepts, such as bits and qubits, entropy, and data compression, seem to run parallel to Shannon's ideas. But there are also some sharp differences, such as superpositions, entangled states, and the quantum no-cloning theorem. It isn't that Shannon's classical ideas of information have disappeared—far from it. They still describe the kind of communication and computing that we directly experience. But now we know of another kind of information, quantum information.

The Two-Slit Experiment

- ▶ The great quantum physicist Richard Feynman once said, “Nobody understands quantum mechanics.” But he also said that all of the mysteries of quantum mechanics are to be found in the two-slit experiment.
- ▶ The setup for this experiment is as follows: We have a source that sends out photons one by one. Each passes through an opaque barrier with two small openings. A bit further on, the photon strikes a screen that is covered with a dense array of photon detectors, rather like the array of detectors in the CCD chip inside a camera. The photon is recorded at just one of those detectors.
 - If we open up just slit 1, then the photon might land anywhere in a wide range of places on the screen. The same is true if we open up only slit 2.



- If we open both slits, though, the distribution of photon landing places has alternating bands of high and low probability called *interference fringes*.



- This result stems from the fact that the photon is in a quantum superposition state of passing through both slits. The interference fringes are produced by that superposition.
- ▶ It is easy to think about this in the wrong way. We naturally assume that the photon goes through one slit or the other. We may not know which slit that is, but nature knows; the uncertainty is all in our minds. But that's not what is happening here.
 - If we were to change the experiment and find out which slit the photon passed through—perhaps by placing super-sensitive non-absorbing photon detection devices beside the slit—then the interference fringes disappear. The superposition is undone. The quantum information is destroyed.
 - The possibility of maintaining a quantum superposition relies on the fact that no measurement (any physical record of any kind) is made. The photon must leave no footprints, no trace of the path it takes. It may pass through glass or bounce off mirrors on the way, but afterward, if we examine the glass and the mirrors, we won't find any evidence of its passing. The photon must be “informationally isolated” from the rest of the world.
 - The ability of an informationally isolated system to maintain its superposition state is called *quantum coherence*. Quantum information survives only while there is quantum coherence. That coherence is very delicate. If even a single atom records the path of the photon, it is lost.
- ▶ As a side note, this is not an all-or-nothing proposition. We could imagine another experiment, in which our super-sensitive photon detectors are not particularly efficient. They give us less than one whole bit of slit information for each photon. When we do that experiment, we find a “diluted” version of the interference pattern.
 - Some coherence has been lost, but some remains. There is a mathematical trade-off between the amount of slit information we gain and the amount of coherence that remains.

- Roughly speaking, every bit of information we gain about the path represents the loss of at least a qubit of quantum information.
- ▶ In some ways, quantum coherence is like a magic trick. You might watch a magician on stage put an egg in a box and close it, wave his magic wand, and draw out a rabbit. If the box were made transparent, you would be able to tell that there was a secret compartment holding the rabbit. But the quantum magic box is different. If we make the box transparent to observe the magic, the trick doesn't happen. It depends on the physical fact that no one is watching. Quantum coherence relies on informational isolation.
 - This is why we never see those effects for the macroscopic objects in our everyday experience. Big things, such as billiard balls and electrical signals in wires, continually interact with other things. Gas molecules and photons bounce off the billiard ball; copper atoms in a wire are agitated by the passing current. These things make records of their passing and cause extremely rapid decoherence.
 - The Universe is a vast network of continual information exchange. But very small things, such as single particles, can sometimes separate themselves from that network for a while. That is when quantum coherence and quantum superpositions can do their magic.

Quantum Error Correction

- ▶ We have already seen a quantum version of data compression—Shannon's first fundamental theorem. Are there quantum versions of other landmarks in Shannon's information theory? For example, we should regard decoherence as a kind of noise. Can we defend against it in the same way that we can defend against classical noise in Shannon's theory? Is there quantum error correction?
- ▶ When quantum information theory was getting started in the 1990s, some researchers believed that quantum error correction had to be impossible. After all, error-correcting codes are based on copying information.

- Think about the simplest error-correcting bit code, the “two-out-of-three” code. The codewords are 000 and 111—three copies of the same bit of information. Even if one of the bits is changed by noise, the majority still indicates the original bit. If we could not copy qubits, surely we could not make a code like that.
- ▶ But in the mid-1990s, Peter Shor of Bell Labs and Andrew Steane of Oxford University discovered the first quantum error-correcting codes. By using many qubits to encode the quantum information of 1 qubit, their codes made that qubit of information more resistant to errors.
 - The qubit state 0 might be represented by the codeword state 000 of 3 qubits. State 1 becomes the codeword state 111. A superposition of 0 and 1 becomes a superposition of 000 and 111. That’s not three copies of the qubit state; it’s an entangled state of 3 qubits. The quantum information is not duplicated; it is shared by all three codeword qubits. And that makes it resistant to error processes that affect only 1 qubit.
 - If the quantum information is collectively held in the entanglement among many qubits, then a failure of information isolation of only 1 qubit cannot quite reach that information. No single qubit has it.
 - This quantum code is a little too simple, but there are longer codes involving 5 or more qubits that are entirely immune to single-qubit errors. Any 1 qubit of the codeword could suffer decoherence or be damaged or destroyed, and still, the original quantum information can be reconstructed by the receiver.

Quantum Information Capacity

- ▶ Shannon’s second fundamental theorem tells us exactly how much error correction is possible—exactly how many bits of information a noisy channel can convey without error. That’s the information capacity of the channel. Unfortunately, we do not yet have the quantum version of the second fundamental theorem. We can define the quantum information capacity mathematically, but we can’t calculate it.

- ▶ In Shannon's information theory, suppose we have two noisy information channels, each with a capacity C . If we use them together, the overall capacity is $2C$. The information capacity adds up. Thus, if we use a single channel many times or many copies of the same channel in parallel, then we can just multiply C by the number of channels.
- ▶ Many researchers thought the same would be true for quantum channels but could not prove it. Then, in 2009, Matthew Hastings found a mathematical example that explained the problem: The assumption wasn't true.
- ▶ Consider two noisy quantum channels. The expression for the maximum quantum information through one channel is Q . When we use the two channels together, we actually can get more than $2Q$. The reason for this is that the channels can be used in an entangled way. The extra capacity in the Hastings example does seem to be very small, but no one has proven that it always must be small.
- ▶ The Hastings example is telling us something subtle and important about quantum channels and their relationship to entanglement. We just don't quite yet know what that message is. Until we do, the long-sought-after second fundamental theorem of quantum information will remain just beyond our grasp.

The Quantum Computer

- ▶ The general idea behind the quantum computer is as follows: An ordinary computer stores its data as bits in its memory. It runs programs that are also stored as bits. A quantum computer, in contrast, stores its data and programs as qubits. That means that its memory can be in superposition states. It can execute superpositions of different programs or the same program on superpositions of different input data. It is almost as if the same computer can do many different computations at once.
- ▶ Unfortunately, we cannot read out all the answers to all those different computations. But in some cases, we can combine the answers using

a form of quantum interference and obtain a single result that no single ordinary computation could provide. This means that for certain mathematical problems, a quantum computer can arrive at the answer in exponentially fewer basic steps than any ordinary “classical” computer. One such problem, as Peter Shor discovered, is the problem of factoring.

- ▶ We saw in Lecture 12 that it is a computationally hard problem to take apart a large integer into prime-number factors. A number with a few hundred digits could not be factored in millions of years on any ordinary classical computer, yet a quantum computer of a few thousand qubits could solve that problem quickly.
- ▶ It’s important to keep in mind that a quantum computer is not magic. It cannot calculate an uncomputable function, such as Turing’s halting function, or tell us the first 1000 bits of Chaitin’s Ω . In that sense, it is no more powerful than a Turing machine. But the line between hard computational problems and easy computational problems is apparently not the same for classical and quantum computers.
- ▶ No one has yet built a quantum computer having more than a handful of qubits because all the computer’s qubits must be informationally isolated from the surroundings. At the same time, they cannot be isolated from each other, because they must exchange quantum information to carry on their computation. That combination is difficult to achieve. At the moment, all the proposals for large quantum computers look like Babbage’s analytical engine: impressive in conception but perhaps beyond the abilities of their inventors to build.
- ▶ If a quantum computer is eventually built, we will face a serious problem. Almost all public-key cryptography today is based on factoring. If quantum computers ever make that into an easy problem, every one of those cryptosystems will become insecure. We will need to discover new ways to keep our secrets. Luckily, quantum information, which poses the problem, may also provide the answer: *quantum cryptography*. Where else should we look for absolute privacy than in the physics that works only when no one is looking?

TERMS

quantum entanglement: A monogamous information relationship between quantum particles, in which neither particle by itself is in a definite quantum state, but the two together do have a definite quantum state.

quantum no-cloning theorem: The information stored in a quantum system cannot be perfectly copied by any physical process. This fact, first proved in 1982, is one of the fundamental differences between quantum information and classical (Shannon) information.

qubit: The basic unit of quantum information; the information contained in a quantum system with just two distinguishable states, such as a photon polarization. In addition to 0 and 1 states, qubits can exist in many different quantum superposition states or in entangled states with other qubits. According to the quantum version of Claude Shannon's first fundamental theorem, the von Neumann entropy of a quantum information source equals the minimum number of qubits necessary to transfer the output of the source.

READINGS

Aaronson, *Quantum Computing since Democritus*, especially chapters 9–10.

Gleick, *The Information*, chapter 13.

Seife, *Decoding the Universe*, chapters 6–7.

Siegfried, *The Bit and the Pendulum*, chapter 4.

QUESTIONS

- 1** Physicist A says, “Quantum physics is more digital than classical physics; for instance, the energy of a light wave comes in discrete photons.” Physicist B replies, “Quantum information is more analog than classical information, since a qubit has a whole continuous set of superposition states besides 0 and 1.” They turn to you for a response. Give it your best shot.
- 2** Why is decoherence much more rapid for large objects than for small ones?
- 3** Do some reading about quantum computers. If you were to take a side of the bottle-of-port bet about quantum computers (made between Laflamme and myself), which side would you take? If we extended the deadline by 20 years, would it change your mind?

Quantum Cryptography via Entanglement

Quantum entanglement between particles has been recognized and studied for a long time. The term goes back to Schrodinger, who called it the “characteristic trait of quantum mechanics.” Albert Einstein and Niels Bohr debated its meaning. In recent decades, we have probed the phenomenon of entanglement with experiments of increasing delicacy and sophistication. Quantum information theory has taught us to think of entanglement as a kind of information. As we will see, we can measure it, transform it from one type to another, and use it for various purposes.

The Basics of Quantum Entanglement

- ▶ The essential idea of quantum entanglement is simple to explain. For a pair of qubits, we can imagine the state 00 or the state 11. The quantum rules say that we can also have a superposition of those: $00 + 11$. That’s a definite quantum state of the pair of qubits, in which neither qubit by itself has a definite state of its own. The von Neumann entropy of either qubit is 1 bit, but the entropy of the whole is 0. That puts the qubits in a special relationship to each other—that’s entanglement.
- ▶ Suppose Alice and Bob are in separate places. They want to produce pairs of entangled particles, one of which is in Alice’s possession and the other in Bob’s. The two do all sorts of manipulations of quantum information at their separate locations. They send ordinary classical messages back and forth. They are allowed to do any local operation or classical communication (LOCC) operations. But try as they might, they cannot create any entanglement between them. LOCC operations are not enough to make entanglement.

- ▶ To achieve entanglement, they must exchange quantum messages—quantum information transmitted coherently, with its superpositions intact. Alice could create an entangled pair of qubits in her own lab and send 1 qubit to Bob. Equally, Bob could create the pair and send 1 qubit to Alice, or they could both receive their own qubits from a third-party source of entangled pairs. In any case, they wind up sharing some quantum entanglement. What can they do then?

- ▶ One answer is that they can change the entanglement from one form to another. Suppose the entangled pairs of qubits are not very entangled. Their superpositions are 00 plus just a tiny amount of 11. There is much less than 1 *ebit* of entanglement in them. By LOCC operations, Alice and Bob can turn a large number of weakly entangled pairs into a smaller number of strongly entangled pairs. This is called *entanglement concentration*, and it is the entanglement version of data compression.

- ▶ The entangled pairs might also be noisy. When Alice sends her qubit to Bob, it might undergo some decoherence. Too much decoherence and the entanglement is lost. But it might be that Alice and Bob's qubits are still entangled but in a noisy way. In this case, by LOCC operations, Alice and Bob may be able to refine their impure entanglement into a smaller amount—fewer qubit pairs—with more pristine entanglement. This is called *entanglement distillation*, and it's the entanglement version of error correction.

- ▶ The preservation of quantum information depends on total information isolation. That means that the quantum world relies on an absolute privacy that is unlike anything in our everyday experience. It's part of the reason that the quantum world seems so strange and elusive to us. And it reaches not just to the behavior of individual particles but to the relationships between particles—to entanglement.

The Quantum Entanglement *Newlywed Game*

- ▶ Let's consider a rather preposterous thought experiment based on an old television show called *The Newlywed Game*. In the game, a recently married couple is separated, and each partner is asked several questions about domestic life, likes and dislikes, and so on. Then, their answers are compared to see if they agree. The results are often embarrassing and amusing.
- ▶ In our version of the game, Alice and Bob will be interviewed separately. We will ask only binary (yes/no) questions. There are only two possible questions, X and Y , and each contestant will be asked only one of them. The questions asked might be the same or different.
- ▶ Because each couple is asked only one question each, if we want to play the game many times, we will need a supply of many couples. We'll assume that each couple has the same "special relationship" to each other. What is that relationship? As far as the game is concerned, it's all about how they answer the questions. It's an informational relationship.
- ▶ If we ask either Alice or Bob either question (X or Y), the answer is completely unpredictable. In probability terms, $p(\text{yes}) = 1/2$ and $p(\text{no}) = 1/2$. On the other hand, Alice and Bob's answers are correlated. If both are asked question X —call that situation (X,X) —then their answers always agree—both yes or both no. For any other combination of questions— (X,Y) , (Y,X) , or (Y,Y) —they always disagree. When one says yes, the other says no.
 - We can summarize Alice and Bob's relationship in a table, listing the probabilities that their answers agree depending on which pair of questions they are asked.

		Bob	
$p(\text{agree})$		X	Y
Alice	X	100%	0%
	Y	0%	0%

- Are Alice and Bob secretly communicating to coordinate their answers? They show no signs of it, and every precaution has been taken, but it is difficult to know for sure.
- We could interview Alice and Bob on different planets at the same moment so that even radio signals, traveling at the speed of light, could not pass between them before they gave their answers. Yet the Alice-Bob relationship, displayed in the probability table, is unchanged.
- ▶ The interviewers in the game, who are information theory experts, devise a mathematical test to see whether Alice and Bob are exchanging secret signals. Here's the idea: If Alice is sending messages to Bob about what happens in her interview, it might be possible to trick her into sending a message for the interviewers.
 - Let QA stand for the question that Alice is asked—either X or Y . And let RB stand for the answer that Bob gives—either yes or no. The question is whether QA has any influence on RB . That is, does Bob's answer give any indication of what Alice is asked? If so, then the interviewers could send their own messages using Alice and Bob.
 - Alice's interviewer encodes his message in the choice of question he poses to Alice, QA . Bob's interviewer would then learn something about the message from Bob's answer, RB . If the interviewers could send a message in this way, that would definitely establish that some kind of signaling is going on.
 - Therefore, the interviewers compute the mutual information $I(QA:RB)$, which as we saw in Lecture 7, is the exact amount of information about QA that appears in RB . The interviewers require as a rule of the game that this must be exactly 0. (This rule is called the *no-signaling principle*.)
 - For Alice and Bob, the exact amount of information about QA that appears in RB is 0. Whichever question Alice is asked, Bob's answer will still be yes or no with equal probability. True, Alice's answer and Bob's answer are correlated, but the interviewers do not control the answers,

so they cannot use them to send messages. And the same thing is true the other direction: $I(QB:RA)$ also equals 0. Bob's interviewer cannot send a message to Alice's interviewer using the game.

- Given that the no-signaling principle is satisfied, the interviewers conclude that there is no evidence of any cheating by Alice and Bob. Either they are not signaling at all, or they are doing so in such a way that they cannot get caught, even in principle.
- ▶ Now, a third contestant, Bill, appears. He claims to have the same relationship with Alice that Bob does. In other words, he claims that his answers to questions X and Y will agree or disagree with Alice's answers in exactly the same way that Bob's do. Let's test to find out if this could be true.
 - We'll interview Bill, Alice, and Bob separately. We will ask Bob question X and Bill question Y . Bob and Bill will give their answers, which either agree or disagree. Suppose the answers agree; what if Alice is asked question X ?
 - Because Bob is asked X and Alice is asked X , then their answers will agree. However, that means that Bill would also have to agree with Alice. But according to the probability table, when both parties are asked different questions, they can never agree.
 - Suppose Bob and Bill give different answers to their questions. Now what if Alice is asked question Y ? Because she is asked Y and Bob is asked X , their answers will disagree. But that would mean that Bill would agree with her—again, a violation of the table. If they are both asked Y , they have to disagree.
 - Thus, either way Bill answers his own question—the same as Bob or different—there is a possible question for Alice that would disprove Bill's claim to have the same special relationship with her that Bob does. Bill must be an imposter!
 - The relationship between Alice and Bob must be exclusive to them. No outsider, anywhere in the Universe, can share in it. It is perfectly and provably monogamous.

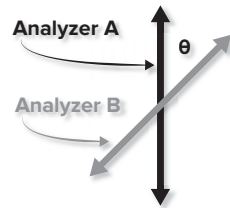
- That is an extraordinary conclusion to be able to draw from a simple table of probabilities. And it does not even depend on Alice and Bob always agreeing or disagreeing. For example, suppose we modify the probabilities in the table so that 100% agreement becomes 85% and 0% becomes 15%, as shown below. Nevertheless, it's still a relationship that would thwart any impostor. No matter what answers Bill gives to his question, he will either end up agreeing with Alice too often or not often enough.

		Bob	
$p(\text{agree})$		X	Y
Alice	X	85%	15%
	Y	15%	15%

Quantum Entanglement and Photon Polarization

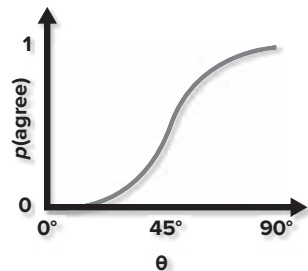
- This game of Alice and Bob is a parable of quantum entanglement. We'll explain it further using photon polarization. Alice and Bob each possess a photon—a qubit. Alice has photon *A* and Bob has photon *B*. Their “special relationship” is quantum entanglement between the photons.
- In experimental physics, it is not actually difficult to obtain entangled photons. We need only a moderately bright laser and a crystal of beta barium borate. When we bombard the crystal with laser light from a particular direction, photons from the laser are converted into pairs of lower-energy photons that zoom off in different directions. Those photons are entangled.
- We can ask questions of individual photons using a polarization analyzer, which can be something as simple as a sheet of polarizing plastic and a photon detector. If we orient the analyzer in a certain way, we are essentially asking the photon, “Are you vertically polarized?” The photon answers, “yes” (vertically polarized) or “no” (horizontally polarized). If we orient the analyzer in a different way, we are asking a different question.

- For each of our entangled photons, the answer to any polarization is random. Yes and no answers are equally likely. But the answers are correlated, and the details of that correlation are significant. If we ask the same analyzer question of each photon—we orient the analyzer along the same axis for each—then the photons always give exactly opposite answers. We ask, “Are you vertically polarized?” If photon A answers yes, then photon B will always answer no and vice versa.



- What if we set the analyzers differently for photons A and B? Suppose we set their axes at an angle θ (theta) apart. Then the probability that the two photons give the same answer follows a simple rule: $p(\text{agree}) = \sin^2(\theta)$. We can understand this more clearly with the graph and table of probabilities shown here.

- When the angle is 0° degrees, $p(\text{agree}) = 0$; the photons never give the same answer. When the angle is 90° degrees, $p(\text{agree}) = 1$; the photons always give the same answer. When the analyzers are 45° apart, $p(\text{agree}) = 0.5$; the photons agree about half the time.

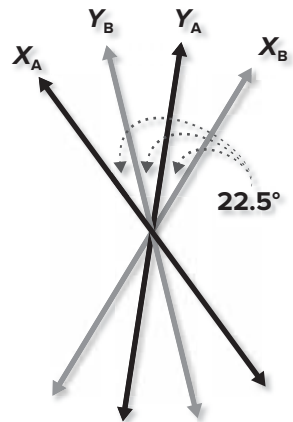


- Two other angles are particularly interesting. At 22.5° (halfway between 0 and 45), the photons agree only 15% of the time. At 67.5° (halfway between 45 and 90), the photons agree 85% of the time.

θ	$p(\text{agree})$
0°	0.00
22.5°	0.15
45°	0.50
67.5°	0.85
90°	1.00

- Here is an important point: Entangled photons satisfy the no-signaling principle. If you want to think that entangled photons are secretly exchanging messages instantaneously across space, you are in good company. Einstein called entanglement “spooky action at a distance.” But you will never catch them at it, and you cannot ever use entanglement to send instantaneous messages of your own. As far as quantum information is concerned, entanglement is not communication but simply an information relationship.

- Now we can set up a *Newlywed Game* photon experiment. All we need to do is to choose some analyzer angles to represent the X and Y questions for the photons, as shown here.



- For photon A, the X analyzer axis points north and the Y axis is northeast. For photon B, the angles are a little more complicated; X is east-northeast and Y is north-northeast. The point is that the angles between adjacent axes are all 22.5° , but X_A is 67.5° from X_B .
- Next, we make a table of “agreement” probabilities for the entangled photons:

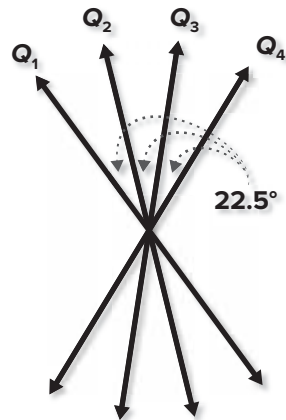
		Photon B	
$p(\text{agree})$		X_B	Y_B
Photon A	X_A	85%	15%
	Y_A	15%	15%

- We saw exactly this same table for Alice and Bob in our *Newlywed Game* thought experiment. Thus, we can draw the same conclusions: The relationship between entangled photons is perfectly and provably monogamous. No other particle anywhere the Universe can share in that relationship.
- ▶ The **monogamy** of entanglement is the deep meaning of the quantum no-cloning theorem.
 - Suppose we have a magic quantum cloning machine, a machine that exactly duplicates the state of a quantum particle. If we begin with entangled photons A and B and use the machine on B, it would have to produce a second photon (B') that entangled with A in exactly the same way as B. But we know that no such photon can exist. Any attempt to create a clone of B must fail. Either the new photon B' will turn out to be unrelated to A, or the cloning process will destroy the original relationship between A and B.
 - To put it in a nutshell: Because of his relationship to Alice, you can't clone Bob!

Secret Key Distribution

- ▶ The unassailable privacy of quantum information and quantum entanglement is a remarkably useful information resource. With it, we can solve an otherwise impossible problem in the science of cryptography: the problem of secret key distribution.
- ▶ As we saw in Lecture 12, there is such a thing as a perfectly secure cryptographic system: the one-time pad, in which a new secret key is used for every message. Key information needs to be random, shared by the users (Alice and Bob), and completely unknown to any eavesdropper (Eve). But since Alice and Bob consume the key as they use the system, they continually need new key information. How can that be distributed to them in such a way that Eve cannot get it, too?

- ▶ If we stick to the world of classical information—Shannon's world—Alice and Bob can't be sure that their key is secure. There is no kind of classical key information that cannot be copied by Eve on its way from Alice to Bob. And that copying process may involve one-way information flow, from the key to Eve, so that no trace of her activity remains behind. That's where quantum physics comes in.
- ▶ The first method for **quantum key distribution** was devised in 1984 by Charles Bennett of IBM and Gilles Brassard of the University of Montreal. Their scheme is known as BB84, and its discovery marked the birth of quantum cryptography. However, BB84 is somewhat elaborate, and it's somewhat difficult at first to see why it does the job. A few years later, Artur Ekert of the University of Cambridge realized that the complications of BB84 obscured the real point of quantum cryptography: In effect, Ekert showed that entanglement is the key. Let's see how this might work.
- ▶ We can produce entangled pairs of photons in the way we have already described and send them off, one to Alice and the other to Bob. Alice and Bob are equipped with polarization analyzers; in other words, they can ask yes/no questions of their photons. Each analyzer can be turned to any of the four axes we used earlier, which we will label Q_1 through Q_4 .
- ▶ When Alice receives her photon, she randomly chooses one of four axes, Q_1 through Q_4 , and finds out if her photon has that polarization. Bob does the same for his photon. Because he is choosing his axis independently, he might make the same choice as Alice or a different one. They repeat this many times, for many pairs of photons.



- ▶ Once they've acquired all their data, Alice and Bob have a discussion on an open classical information channel, that is, a discussion that Eve might hear. They tell each other which measurements they made on which photons, but they do not announce the results of those measurements.
- ▶ On about 25% of the photon pairs, they chose the same analyzer axes. On these entangled photons, that means that their results must disagree. It's easy for Bob to negate his results, so that they always agree. Now Alice and Bob each have a collection of random, shared, and completely secret bits—in short, they have a perfect new key.
- ▶ Even if Alice and Bob are suspicious of the source of the entangled photon pairs, they can still achieve complete confidence in the secrecy of their key.
 - Remember, 75% of the time, they chose different axes for their two photons. These are not as good for building a secret key because they are not perfectly correlated.
 - Instead, Alice and Bob share these results with each other over their open channel. By comparing notes, they can test the probabilities in the *Newlywed Game* experiment. They can test that Q_1 and Q_2 results agree only 15% of the time, that Q_1 and Q_4 results agree 85% of the time, and so on.
 - From this data alone, they can establish beyond any doubt that the information relationship between their photons is completely monogamous. No matter what tampering Eve may have attempted, Alice and Bob know that she can possess no particle or measurement record that shares their key information.
 - If Eve did attempt to do something that might reveal key information—by using the photon pair to create a third photon for herself, for example—her activities would necessarily destroy the monogamous connection between the photons of Alice and Bob, and they would detect this when they compared results.

- Thus, Eve can interfere with the key distribution process, but she cannot fool Alice and Bob into using an insecure key to send their secret messages.
- ▶ Unlike quantum computing, quantum cryptography has gone far beyond a theory and a handful of small experiments. Pairs of entangled photons have been shared over 100 kilometers apart, first using optical fiber links, and later, directly, across free space.
- ▶ Quantum key distribution has also gone commercial. There are several companies that now build and sell quantum cryptographic systems. Some electronic banking transactions in Europe have already been encrypted using key bits shared via quantum key distribution.
- ▶ Most modern cryptography avoids the key distribution problem by using a public-key system. The encryption key is not particularly private—it can be published in the newspaper—but the separate decryption key remains secret. Nevertheless, the security of a public-key system is only computational, not informational. In principle, any eavesdropper has all the necessary data to break the code. The secrecy of the plaintext relies on the sheer computational difficulty of that task. It's equivalent to solving some difficult mathematical problem, such as factoring a large integer.
- ▶ Someday, perhaps not that far off, quantum computers could make factoring a far easier problem. If that happens, the public key systems we use today will become insecure. Worse, all of today's encrypted message traffic, recorded and stored somewhere, will become readable by any adversary with a quantum computer. For secrets that need to be kept over long time scales, one-time pad encryption based on quantum key distribution provides the only real guarantee of cryptographic security.
- ▶ Quantum entanglement and quantum cryptography are perhaps even more valuable because of what they teach us about what information means in our Universe.

- Alice and Bob live in a complex world full of interacting particles. The noisy environment that surrounds them, the dense and intricate web of information exchange, is continually nibbling away at every form of informational isolation.
- In such a world, we may imagine three basic information tasks that Alice and Bob might wish to perform: (1) transmitting quantum information—qubits—from one to the other, defending against the noisy environment by quantum error-correcting codes; (2) sharing some entangled pairs of qubits with each other, qubits whose information relationship is provably monogamous; and (3) transmitting classical messages that are absolutely private—completely secret from any enemy their environment might contain.
- Over the years since quantum cryptography was invented and improved, we have begun to understand a surprising truth. These three things—quantum communication, entanglement, and perfect cryptography—are, at root, the same. If you are able to do one of them, you can also do the other two. Therefore, the science of secrets—the shadowy art of cryptography—is not a mere secondary chapter in the science of information. It is actually the key to unlocking what information means in the quantum world.
- We have also learned another amazing truth. When we showed why the information relation between entangled particles had to be monogamous, we did not actually use the theory of quantum mechanics at all. We used only the observed correlations between entangled photons and the no-signaling principle. This means that quantum key distribution is secure, even if quantum theory itself is wrong. Logically, quantum cryptography is more firmly established than quantum theory itself.

TERMS

monogamy: The exclusive nature of the quantum information relationship known as *entanglement*. If quantum systems A and B are entangled, then no third system, C, can share in this exclusive relationship. The monogamy of entanglement is the basis for the security of quantum cryptography.

quantum key distribution: An entirely practical and commercially available basic protocol of quantum cryptography by which a reliably secret random key can be shared between two parties. The best techniques use quantum entanglement. Any attempt by an eavesdropper to determine the key will necessarily disturb the shared quantum information in a way that the two parties will surely detect, leading them to reject the key they construct as insecure.

READINGS

Aaronson, *Quantum Computing since Democritus*, chapter 12.

Aczel, *Entanglement*.

Single, *The Code Book*, chapter 8.

QUESTIONS

- 1 Is there anything analogous to monogamy of entanglement in Shannon's information theory?
- 2 In quantum cryptography, Alice and Bob use entanglement to establish a secret key, which they then use in an ordinary one-time pad to encode their messages. Why don't they use the entanglement to transmit their message directly?

3 Below are three information resources that Alice and Bob might have. Which of these can be created using the others?

C: They send classical bits of information to each other in public.

E: They share some quantum entanglement.

K: They share some common secret bits to use as a secure cryptographic key.

It from Bit: Physics from Information

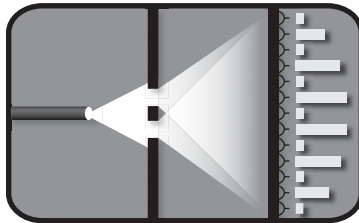
LECTURE 23

John Wheeler was an American physicist famous for his bold ideas and insights about the deepest problems in physics. To a physicist, Wheeler said, the laws of physics are mathematical equations, but even the greatest mathematical theory of the cosmos is missing the principle by which it can become real. Wheeler did not claim to know what that could be, but he thought there were clues scattered throughout the fundamental theories of physics. One important clue, he believed, came from quantum theory and the central role that information plays in it. Always ready with a memorable phrase, Wheeler called his idea “it from bit.”

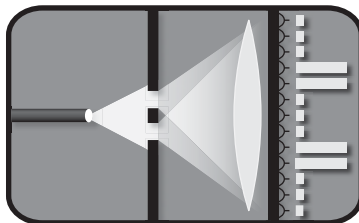
The Elementary Quantum Phenomenon

- ▶ As we saw earlier, depending on how we set up the two-slit experiment, we can either observe the quantum interference or find out which slit the photon passes through. Bill Wootters and Wojciech Zurek, about the time they were proving the no-cloning theorem, also analyzed the two-slit experiment. They showed that there is a strict information trade-off: The more “which slit” information we obtain, the less the quantum coherence leading to the interference effect.
- ▶ Wheeler noted that our own choice of experiment helps to determine which phenomena become real in the two-slit situation. We are not merely detached observers but participants in the process. If we do not observe the slits, quantum interference is real, but there is no answer to the question, “Which slit?” If we do observe the slits, that question does have an answer, but the interference effect is gone. Wheeler also pointed out something even more unsettling: We can wait to decide which experiment we are doing until the last moment—when the experiment is 99% over.

- If we want to see interference, we do the usual experiment, as shown below:



- If we want to determine which slit the photon went through, we can insert a lens just before the screen. The lens deflects the light waves and forms an image of the slits on the screen. The photons now land in one of two spots, corresponding to the two slits.



- The experimental apparatus might be very long, and we can decide whether or not to insert the lens at the very last nanosecond.
- According to Wheeler, we can choose whether or not the photon passed through a definite slit long after the photon went through. He called this a **delayed-choice experiment**. The experiment is not over until the photon is registered by one of the detectors at the screen, which produces an electrical impulse (a click) that can be recorded and studied.

- ▶ Niels Bohr said that an “irreversible act of amplification” is necessary, bringing things from the microscopic realm to the macroscopic realm, so that the result of the experiment can be “communicated in plain language.” Until that happens, we cannot ascribe too much reality to the photon. Wheeler put it this way: “No phenomenon is a phenomenon until it is an observed phenomenon.”
- ▶ The quantum information in that microscopic world—that private world of superposition, interference, and entanglement—must be converted into classical information—Shannon information—which can be copied, processed, and shared with the rest of the Universe. Only then can we say that things have become unambiguously real.
- ▶ Wheeler called it the *elementary quantum phenomenon*—this chain of quantum events that leads to the creation of unambiguous information, information expressible as 1s and 0s. The elementary quantum phenomenon need not be localized in space or time. It stretches from the source of the photons to the click of the detector. It seems like a small thing, but the world is made of a stupendous number of such elementary phenomena: not particles and forces, not fields, not even space and time, but information—“it from bit.”

Black Hole Entropy

- ▶ If the Universe is made of information, how much information does a piece of the Universe contain? This question was asked by Jacob Bekenstein.
- ▶ Suppose we have a sphere that is 1 meter in radius. How much information can we stuff into that sphere? However much it is, common sense suggests that the maximum storage capacity of a region of space depends on its volume. More volume means more room for particles to encode the information. A sphere of twice the radius—2 meters—should be able to store eight times as many bits. But Bekenstein said that this is wrong.

- ▶ To store more and more information in the sphere, we have to add more and more energy. If we put too much energy in, the equivalent mass of the sphere will be large enough to form a **black hole**.
 - A black hole, of course, is a star or other large mass that has collapsed under its own gravity until nothing, not even light itself, can escape from it. It is surrounded by a mathematical surface called the *event horizon* that can be crossed by information only inward, never outward.
 - Thus, once we have made a black hole out of our data storage region, the information is lost, and that's what limits us.
- ▶ Bekenstein had previously made an important discovery about black holes—that they have a thermodynamic entropy. The entropy is given by Boltzmann's formula, $S = k_2 \log_2(M)$, but the number M , instead of counting possible microstates, counts possible histories. That is, it represents how many different ways—from how many different configurations of matter and energy—the black hole could have formed. That is exactly the information that has been lost in the hole.
 - Bekenstein showed that black hole entropy is actually proportional to the surface area of the event horizon. That is, $S(\text{black hole})$ is a constant multiplied by $A(\text{horizon})$. The exact value of the constant was later worked out by the English physicist Stephen Hawking.
 - If we think of black hole entropy as bits of lost information, each bit occupies an area of 7×10^{-70} square meters. A black hole the mass of our Sun is 3 kilometers in radius. Its lost information is more than 10^{77} bits.
- ▶ Based on black hole entropy, Bekenstein concluded that the information storage capacity of any region of space (the maximum number of bits that can be stored without forming a black hole) is not limited by its volume but by its surface area. The limit is 7×10^{-70} square meters per bit. This is an almost inconceivably small area.

The Black Hole Information Problem

- ▶ Let's look at the life cycle of a black hole. It starts out as matter—the collapsing remnant of a massive star. The hole forms, and all the information encoded in that matter disappears behind the event horizon. After things settle down, the black hole emits radiation for a long time. Very slowly, it loses mass and shrinks, gradually evaporating away. Eventually—the exact details of the final stage are not well understood—the black hole is gone. Here's the question: Where did the original information go?
- ▶ It crossed the event horizon into the black hole, but eventually the black hole goes away. There is some very faint quantum radiation discovered by Hawking, but that seems to have nothing to do with the material that went into the black hole. This is called the **black hole information problem**, and it is an especially knotty puzzle in theoretical physics.
- ▶ There are two possible answers.
 - First, the information might simply be lost in the black hole. The leftover radiation is random and independent of the original material.
 - The second answer is based on the principle of microstate information, which states that microstate information is never truly lost. At worst, we lose access to it. This answer, therefore, says that the emitted Hawking radiation carries, perhaps in scrambled form, all the details of the exact configuration of matter that formed the black hole in the first place. The information is never actually lost; it goes into the black hole, only to be re-emitted in another form quadrillions of years in the future.

The Holographic Principle

- ▶ Regardless of the answer to the black hole information problem, the connection between information and area is very strange. Why should the information content of a region of space be limited by its surface? Perhaps it indicates something about the nature of information itself.

- ▶ We have spoken about information as if it were an intrinsic property of an object: This photon carries 1 bit of information; that computer contains such-and-such data. But Wheeler’s elementary quantum phenomenon suggests that information becomes unambiguously real only when it can be communicated and shared with the rest of the Universe.
 - Consider quantum entanglement, which is not really communication but, rather, an information relationship between particles.
 - Thus, perhaps the right way to think of information is as a relation, not an intrinsic property. Two particles are related to one another; one part of the Universe is related to all the rest.
- ▶ This relational idea accords with our intuitive concept of information. We say that a message carries information, not just because of its contents, but because those contents are correlated with something in the outside world. The message is about something!
- ▶ If information is fundamentally relational, then it makes sense that it is limited by surface area. That surface is the boundary between the inside and the outside, between a region of space and the rest of the Universe. The information relationship must reach across that boundary, and there is only so much room. It takes 7×10^{-70} square meters per bit of relation.
- ▶ The logical endpoint of Bekenstein’s ideas is a speculative theoretical concept known as the ***holographic principle***. This was proposed by the Dutch physicist and Nobel laureate Gerard t’ Hooft. The holographic principle states that we can regard the information in any region of space as existing on the boundary surface of that region.
- ▶ Earlier in this course, we discussed the information in a jar of air—microstate information about the air molecules. Add to that information about the light and gravitational fields in the jar, and there seem to be a great many bits in the jar. But t’ Hooft says that it is equally valid to think of those bits as existing on the jar’s surface. After all, any information exchange with the interior—when we look inside the jar and so on—is always mediated by that surface. Our picture of things in the interior is a reconstruction based on those exchanges.

- It isn't that the interior of the jar is an illusion, exactly. The two ideas—information in the interior and information on the surface—are equally valid, complementary pictures of the reality. And they are equally valid for any region of space.
- The world is, like a hologram, a little less three-dimensional than it appears.
- ▶ We cannot say yet where the holographic principle will lead. Many physicists today regard it as the key to a deeper kind of physics. Wheeler always regarded black holes as another clue to something missing in physics. If the elementary quantum phenomenon is the birth of information, the black hole, perhaps, is its death.

The Information Content of the Universe

- ▶ What is the information content of the entire Universe? If information is truly relational—if it is about the relation of the inside to the outside—then the holographic answer for the whole Universe, which has no outside, is 0! Yet there is an interesting calculation to be made.
- ▶ Just as a black hole is surrounded by an event horizon, so we, in our expanding Universe, are surrounded by a cosmic horizon. Think of it as a tremendous sphere, billions of light-years in diameter, with us at the center. Events beyond that horizon are not visible to us—or not yet at least. The Universe has not existed long enough for information from those regions to have reached us.
 - The horizon grows over time; we see more and more of the Universe. But at any given moment, the observable part of the Universe is finite, and the cosmic horizon is its boundary.
 - If we divide the area of that surface by the tiny area for 1 bit, we get a huge number: about 10^{120} bits.
- ▶ In 2001, Seth Lloyd from MIT tried to figure what such a number means. He said that we can regard the Universe as a gigantic information-

processing system. In effect, the Universe is a computer, and its evolution is its computation. We can try to figure out how big that computation is: How many bits are involved? How many basic logic operations? This would be an interesting number, especially if you hold the rather fantastic idea that our Universe is a simulation running on a computer in some higher, unimaginable realm. This calculation would tell us how big that computer has to be to handle the simulation.

- ▶ The cosmic horizon number, 10^{120} bits, is possibly too large because it should give the maximum amount of information that can be contained within a volume the size of our Universe. But the Universe is mostly empty. Its information content is less.
- ▶ Lloyd said that we can estimate the number of bits in the cosmic computation by estimating the total entropy of the contents of the Universe. The lion's share of that entropy is actually not in stars or gas but in the cosmic microwave background, the faint radiation left over from the hot, dense era soon after the big bang. The entropy of that radiation is about 10^{90} bits. That's much smaller, but it is still gigantic.
- ▶ Lloyd next turned to estimating the number of basic computer operations that the Universe has executed since the big bang, counting in "ops" rather than bits.
 - He noted that energy is required to do any computation. That energy need not be lost as waste heat—the computation can be thermodynamically reversible—but energy must be present for the operations to take place, and the more energy there is, the more rapidly those operations can happen.
 - Armed with this idea, Lloyd estimated that the total number of basic computer operations on the 10^{90} bits in the Universe has been about 10^{120} ops—the same as the bit number for the cosmic horizon.
 - In fact, Lloyd showed that this is no coincidence but would hold true in any expanding universe like ours. The horizon bit number equals the cosmic op number.

- ▶ This sort of calculation can be a little dizzying, and it raises some unsettling questions. For instance, Lloyd's calculation suggests that the Universe is fundamentally finite. It contains a finite amount of information and has performed only a finite number of basic logic operations. The numbers involved are large, but even a large number is very far from infinity.
 - Mathematics, in contrast, is all about the infinite: There are infinitely many points on a line. The irrational number π has infinitely many decimal places.
 - Yet every computation we do is done within the Universe. It is a small piece of the cosmic computation and is, therefore, limited by the cosmic limits.
 - That being true, what does it actually mean to say that π has an infinite number of decimal places? Almost all of those digits can have no meaning and no existence in the physical world.

The Minimum Logical Basis for Quantum Mechanics

- ▶ The quest for an information basis for physics has recently gone in an interesting new direction. The idea is to try to identify where quantum mechanics itself comes from. Perhaps we can answer this question by figuring out the minimum logical basis for quantum mechanics: What is the smallest set of postulates, of the simplest kind, that would lead to the whole quantum mathematical machinery? Can we derive quantum mechanics from axioms about information?
- ▶ Mathematical physicists have found a number of lists of axioms with an information "flavor" from which the whole theory of quantum mechanics can be derived. There are usually about half a dozen axioms in a set, all but one of which is very straightforward. The axioms include the no-signaling principle, the idea of data compression, the ability to assemble two or more systems into one larger system, and so on. The list varies, but all of the axioms are fairly intuitive. They're also perfectly consistent

with classical physics and ordinary Shannon information. Then there is always one more axiom, the “special sauce,” that turns everything into quantum mechanics.

- For Lucien Hardy at the Perimeter Institute in Canada, the special sauce is the idea that we can go continuously from one information state to another. That isn’t true in Shannon’s theory. We have to jump from 0 to 1. But a qubit has superposition states between 0 and 1, and that allows the change to be continuous.
- For the Italian physicists Giacomo D’Ariano, Giulio Chiribella and Paolo Perinotti, the special sauce is the idea that any noisy state of a system is really just part of a pure entangled state of a larger system. Entropy is always about entanglement.
- In addition to these special axioms, there are others. The hope is that by identifying the essential information principle that determines quantum theory, we will gain an understanding of the information basis of the quantum world. We will be on our way to “it from bit.”

The Connection between Information and Physics

- We cannot at present give good answers to Wheeler’s deep questions with which we started this lecture, but the connection between information and physics has indeed led us in some mind-bending directions.
 - The whole subject of information and physics really had two godfathers, two people whose ideas and influence shaped the work of thousands of others over the last few decades. They were Rolf Landauer and John Wheeler. Their approaches were very different.
 - Landauer said that information is physical. He pioneered the idea of using the laws of physics to understand how information is acquired, processed, stored, and erased in the physical world.

- Wheeler said that physics is informational. He urged us to dig beneath the elegant mathematical laws to uncover the true foundation beneath. He believed that foundation was built from information.
- ▶ Many researchers have conducted work of the Landauer variety. They have tried to understand the concepts and laws of information that arise from quantum mechanics. The whole of quantum information science—quantum communication, entanglement theory, quantum cryptography, quantum computing—is exactly that quest. And these researchers have discovered some amazing things about the physics of information.
- ▶ Yet a surprising number of physicists are motivated by Wheeler’s vision. For them, every new discovery—from the delayed-choice experiment to black hole entropy to the holographic principle—confirms the central place of information in the fundamental laws of nature and brings us a little closer to the dream of “it from bit.”

TERMS

black hole: A massive object whose extreme gravitation prevents even light from escaping. The hole is surrounded by a mathematically defined surface called its *event horizon*; no information from within the horizon can get outside. However, thanks to quantum mechanics, black holes are not entirely black: They emit a very faint thermal radiation. Black holes have an entropy proportional to their horizon area. This entropy represents the information that has been “swallowed” by the black hole.

black hole information problem: Does information that falls into a black hole ultimately disappear from the Universe, or does it eventually reemerge in the quantum radiation from the hole? The subject of a famous bet, in which physicist John Preskill, later joined by Stephen Hawking, argued that the information eventually returns. Kip Thorne was reportedly not yet convinced.

delayed-choice experiment: A type of quantum two-slit experiment on a photon, devised by John Wheeler. We can either obtain “which slit?” information or observe interference effects but not both in the same experiment. In the delayed-choice version of the experiment, the decision about which experiment to perform can be put off until long after the photon has passed slits. Thus, the elementary quantum phenomenon that the experiment represents is not completely localizable in space and time.

holographic principle: A speculative principle of physics based on the work of Jacob Bekenstein and developed by Gerard t' Hooft, positing that the information contained in any region of space can be equally regarded as existing on the boundary surface of that region—the interior of a region is a kind of “hologram” produced by the surface.

READINGS

Seife, *Decoding the Universe*, chapters 8–9.

Siegfried, *The Bit and the Pendulum*, chapters 9–12.

QUESTIONS

- 1 Back in Lecture 16, we estimated the missing microstate information in a liter of air to be about 6×10^{23} bits. If the volume is cubical, each square face has an area of 10^{-2} m^2 . If we envision this information spread over the cube's surface, how many square meters corresponds to 1 bit? How does this compare to the ultimate limit, $7 \times 10^{-70} \text{ m}^2$ per bit?
- 2 What do you think about the research program of finding good information axioms for quantum theory? Do you agree with Wheeler's jibe about axiomatization?
- 3 What do you find to be the most intriguing question or speculation about the relation between information and the laws of nature?

One of the most astounding features of the science of information is its ubiquity. Claude Shannon's revolution has spread far beyond the technology of telecommunication. The concepts and principles he formulated have become essential elements of sciences from biology to quantum physics. Why? What is it about information that makes it such an indispensable idea?

Isomorphism

- ▶ At root, information is about isomorphism. This term, borrowed from mathematics, means that two different mathematical structures are essentially alike. In topology, for example, the shape of a doughnut is isomorphic to the shape of a coffee cup. That's because each one can be transformed into the other in a continuous way. Topologically, each one is a *torus*—the mathematical name for the invariant common quality that those different shapes share.
- ▶ The transformability of shape should remind us of the protean transformability of information. When you speak, your voice is a sound. But later, it can become a sequence of electrical signals, or a static magnetic pattern on a computer disk, or an electromagnetic wave propagating through space. In the end, it is regenerated as sound once more. *Information* is what we call that invariant common quality that all those vastly different physical phenomena share. It is the isomorphism between them.
- ▶ That's why information theory is so perfectly suited to find common ground between language and thermodynamics, DNA and MP3, photons and neurons. And if we wear our "information-colored glasses," we will see the laws of information at work all around us, in hundreds of different ways.

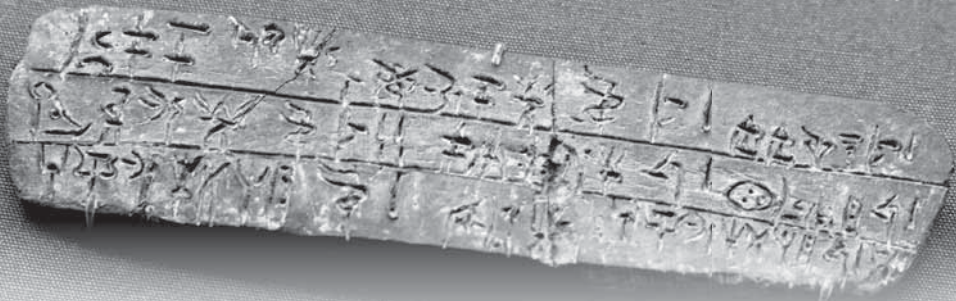
Information and Economics

- ▶ Economics is one of the most remarkable branches of the science of information. Indeed, money itself is a form of communication—a kind of information. This is actually not that hard to believe nowadays, when most money exists in the form of electronic account data—bits in banks—but it has been true from the beginning.
- ▶ Money was invented to solve a problem. If you have fish and someone else has baskets, you can agree to trade. Three-way trades are more complicated to arrange: You give someone fish, she gives someone else baskets, and he gives you a spear. When time is involved, it's even worse: fish today for baskets tomorrow for spears delivered last week. Money is a sophisticated record-keeping and communication system for all those agreements and obligations, direct and indirect, that we undertake.
- ▶ We should not be distracted by the fact that early monetary systems used coins made of precious metal. The real purpose of that was cryptographic: to make it hard to counterfeit the messages of agreement and obligation. If we can meet that cryptographic requirement, we can use paper money or digital money. The very transformability of money tells us that money is information.
- ▶ On the other side, prices—the exchange rates between money and goods—are a very sophisticated system of communication, distributing information to widely separated people who need to know it.
 - An earthquake in Indonesia that shuts down the tin mines will affect the future availability of tin. That will have implications for people all over the world who use tin, whether or not they know it. How do they get the word? The price of tin goes up.
 - Users respond by finding ways to do with less or by substituting something else. They raise their own prices to pass the message along.

- Yet the ball-bearing manufacturer in Germany and the cell phone user in Brazil know nothing about the Bangka Malay tin miners. They do not share a language or a culture. But they can communicate because they are part of the same network of exchange.
- At each link in that network, people understand each other well enough to buy and sell. That's all that is necessary.
- ▶ The concept of information has become central to modern economics. More than one Nobel Prize in Economics has been awarded for studies of information asymmetry—the situation in which one party to a transaction has access to more information than another. Economists try to discover ways to make that economic information network more efficient and effective. And more recently, even the nitty-gritty of Shannon's information theory has begun to find its way into some economic thinking.
- ▶ People take actions and adjust prices based on the information available to them, but real human beings cannot process an unlimited number of bits of data. People have a finite information capacity and will simply miss some signals. Recent computer models suggest that this “information capacity” idea can help to explain the long-observed fact that price changes often lag well behind the changes in circumstance that drive them. That's called *price stickiness*, and it is at least partly due to the fact that we economic actors can acquire and use only so many bits per day.

The Problem of Anti-Cryptography

- ▶ All communication, economic or otherwise, is based on what we share. That makes sense because all communication is based on codes. A code is an arbitrary rule that associates a symbol with a message. In the absence of that shared rule, a collection of symbols could literally mean anything.



LINEAR B SCRIPT

- ▶ Ventris and Chadwick were able to decipher Linear B because they recognized the Greek language in symbols scratched on clay. They shared a common code with the writers of the tablets—spoken Greek—and that was enough to break the written code. Linear A is an older writing system from Crete that uses some of the same symbols as Linear B. But the ancient Minoans who wrote it did not speak Greek; in fact, we do not know what language they spoke. Thus, Linear A remains an unsolved puzzle to this day. The Minoans cannot speak to us.
- ▶ What if we want to communicate with people thousands of years from now? How could we hope to be understood by people so remote from us that we would share no common code with them? The physicist Philip Morrison has called this the *problem of anti-cryptography*. In the absence of a shared code, how can we not keep our message secret?
- ▶ Actually, sending information even 200 years into the future (in a time capsule, perhaps) poses some challenges. The question is not basic understanding—it is easy to read books from 200 years ago. The problem is that most of the information we have and use today is electronic in form, but most of the technologies we use to store that information won't last for 200 years. For our time capsule, the best bet is to use old-fashioned data storage technologies, such as print on acid-free paper or inscriptions on metal or stone.

- ▶ On the longer time horizon, we face the problem of language. Languages change over time. Each generation speaks a little differently from its parents. We can reconstruct the history of language change by studying and comparing many different languages and dialects. That allows us to build a “phylogenetic tree” for human languages. However, even with plenty of linguistic data, there seems to be no way to reconstruct languages back more than about 6000 years. Things simply change too much over that span. How, then, will we transmit information to our remote descendants, 10,000 years from now?

Communicating with Our Descendants

- ▶ One practical reason we might want to communicate with our descendants involves nuclear waste stored in the Waste Isolation Pilot Plant (WIPP) in New Mexico. This facility was built to demonstrate how nuclear waste can be permanently stored in geologically stable areas. However, the waste stored there now will not decay to the level of natural uranium ore for 10,000 years. Until then, we would like to tell our descendants to leave the area alone.
- ▶ This problem has been studied by several teams of scientists, with fields of expertise ranging from nuclear physics to anthropology. They pondered what kind of monument or inscription could carry the WIPP warning message far into an unknown future.
- ▶ We cannot rely on ordinary written language or other kinds of symbols. For example, the trefoil, the international symbol for radiation, may lose its current meaning. Even the skull-and-crossbones, the symbol for toxicity, could be misinterpreted. And we can make no confident predictions about the science and technology of future humanity. They may be far more or less advanced than we are.
- ▶ The scientists who studied the problem came to realize that the solution involved a multilayered message, each layer building on the content of the last. The first layer is straightforward: “People made this.” The second layer is also simple: “Stay away from here.” The third layer is

more ambitious; it explains in a general way that something toxic is buried at the site and that it will not be safe to dig it up for 10 millennia. The fourth layer adds more technical detail, explaining the design of the facility, the nature of the waste, and so on.

- ▶ The plan is that WIPP will be surrounded by an earthen berm, 30 feet high and a half-mile across. Buried in the berm will be magnetic markers and metal targets visible to radar. In the middle will be a roofless granite building with inscriptions and two more buried information vaults. Smaller stone markers will be buried at various depths around the site. Finally, extensive technical records about WIPP will be deposited in archives all over the world, in hopes that some copies may survive.
- ▶ The outer layers of the message will have to be largely pictorial. One cartoon might show someone digging, then growing very ill and dying. A star chart could convey the notion of 10,000 years by using the slow precession of the Earth's rotation axis. Diagrams could convey mathematical concepts, such as numbers, which could in turn be used on blueprints and technical data. Small samples of the entombed waste would likely be buried at much shallower depths to permit scientific study.
- ▶ The danger to guard against is the assumption that the recipients share parts of our many visual or linguistic codes. For example, we might consider the meaning of an arrow pointing in a certain direction to be obvious, but they might not.
 - Nevertheless, we can reasonably expect that we will share some things with the future recipients of our warning message. Physical features of the environment will stay relatively constant; the constellations will shift but not noticeably change shape; and so on.
 - Most importantly, the recipients will be human beings, like us. Studies of many different human cultures will allow us to build on human universals, such as the ability to recognize the human form in various actions or the human face showing the emotions of happiness or distress. These universals are shared information between us and future humanity and can form the basis of a visual code for speaking to our distant descendants.

- That's how we can solve Morrison's anti-cryptography problem—by making a code out of what we naturally share. With some effort, we have a decent chance to transmit our warning and have it understood.

Communicating with Extraterrestrial Intelligence

- ▶ The notion that there could be thinking beings on other worlds has been around since antiquity, but modern research into this idea began in 1959 with a paper entitled “Searching for Interstellar Communications” by Philip Morrison and the Italian physicist Giuseppe Cocconi.
- ▶ Cocconi and Morrison calculated that existing radio telescopes were sensitive enough to detect signals from their counterparts orbiting nearby stars. Those stars and the background of other radio sources in the galaxy would contribute noise. To achieve a signal-to-noise ratio of at least 1 (so that the signal would be easy to detect) the bandwidth of the receiver would have to be very narrow. The bit rate would be low. But if extraterrestrials existed within a few dozen light-years of Earth, had radio telescopes as advanced as ours, and happened to be beaming signals in our direction, we could hear them. Thus, Cocconi and Morrison proposed that we listen.
- ▶ Two years later, Frank Drake, a radio astronomer at Cornell University, used a radio telescope at the National Radio Astronomy Observatory in West Virginia to monitor a pair of nearby stars for possible intelligent radio signals. The stars he chose were two of the favorable candidates that Cocconi and Morrison had listed, and he monitored close to a wavelength they had suggested as a natural “landmark” in the radio spectrum. Drake did not discover alien radio signals, but his experiment—called Project Ozma—was the first step in the scientific search for extraterrestrial intelligence (SETI).
- ▶ In 1974, Drake was director of the radio telescope facility at Arecibo in Puerto Rico, the largest radio dish antenna on the planet. The telescope had been upgraded with a powerful transmitter for interplanetary radar work. To demonstrate the new capability, Drake and the Arecibo team



decided to turn SETI upside-down. Instead of just listening for deliberate alien signals, we would send a signal of our own. Drake got his Cornell colleague, the astronomer Carl Sagan, to help him design the message. The result is one of the most fascinating examples of anti-cryptography ever devised.

- The **Arecibo message** was a sequence of 1679 binary digits encoded in slight shifts of the radar frequency—FM—and transmitted at 10 bits per second. The number 1679 is the product of two prime numbers, 73 and 23. Thus, a recipient might get the idea to arrange the bits in a rectangle, either 73 by 23 or 23 by 73. From one direction, the result is apparent nonsense, but from the other direction, a picture appears, as shown in **Figure 24.1**.

- There is no way to tell how one should orient the picture. An alien recipient might have it flipped horizontally or vertically. But the first bits transmitted appear in the upper-left corner, so that is the place to begin.
- The top row is a lesson about how to count to 10 in binary. Each number includes a bit to mark the end of the number. The rest of the sequence is familiar: 1, 10, 11, 100, 101, and so on. The last few numbers (8, 9, and 10) demonstrate that we may put extra bits to the side for longer numbers.



- The next item is a group of five numbers: 1, 6, 7, 8, and 15.

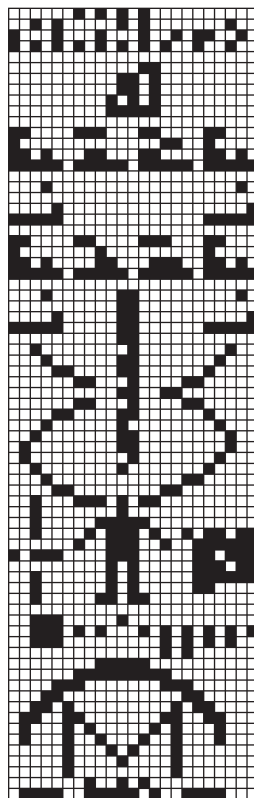
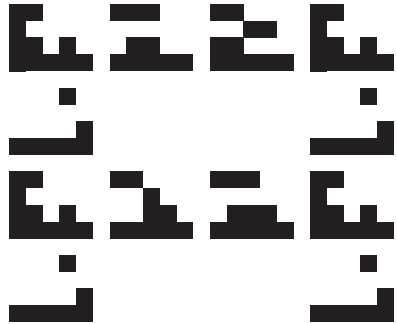


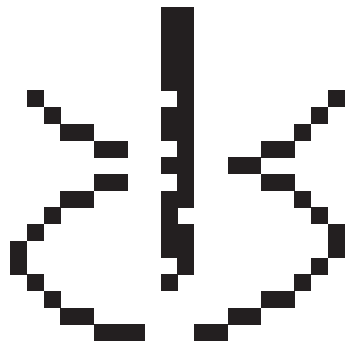
Figure 24.1

These are the atomic numbers for five elements: hydrogen, carbon, nitrogen, oxygen, and phosphorus, the elemental building blocks of life on Earth. That might seem a little too obscure, but we must assume that the recipients know a good deal about physics and chemistry and would be bound to hit the right interpretation eventually.

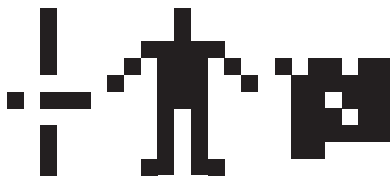
- After this comes an arrangement of five-number groups. Having figured out the atomic numbers, the recipients could deduce that these represent chemical formulas. The first is seven hydrogens, five carbons, and one oxygen—a sugar. The next one down is four oxygens and one phosphorus—a phosphate. Then the same two units repeat, just as they do on the other side. That's a description of the twin sugar-and-phosphate backbones of the DNA molecule. And the formulas in the middle represent the four DNA bases: adenine and thymine above, cytosine and guanine below.



- This part of the message is a compact description of the molecular storage system for our genetic information, the most fundamental basis of Earth life.
- The next part of the message shows a rather impressionistic picture of the double-helix structure of DNA; in the middle of it is a large binary number. We can tell it's a number by the small bit marker at the bottom. The number is 32 bits long, which translates to about 6 billion—the approximate number of base pairs in the human genome.



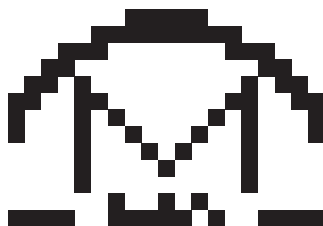
- Below the genome number there is a picture of a human, and beside the human are two numbers. Each is to be read horizontally rather than vertically, as suggested by the bit marker. The number on the right is a 32-bit number equal to about 4 billion, the approximate human population of the Earth in 1974. The number on the left, in the middle of the vertical bar next to the human, is a measure of our height: 14. This represents the only unit of length that we are certain the aliens must share with us: the wavelength of the radio wave that carries the message. The Arecibo radar transmitter had a 12.6-cm wavelength; multiply that by 15, and we get 176 cm—about 5 feet, 9 inches.



- Next there is a diagram of our Solar System, the Sun and nine planets (including Pluto). The larger planets are indicated by longer bars. The third planet, Earth, is slightly displaced toward the human in the diagram.



- At the bottom is an explanation of how the signal was transmitted: a diagram of the Arecibo antenna. The width bars and binary number below give the antenna's diameter: 2430 wavelengths, or a little more than 300 meters.



- ▶ The Arecibo message was transmitted on November 16, 1974, toward the dense globular cluster M13, about 22,000 light-years away. Given that the signal was broadcast only once, it is unlikely that anyone in M13 will happen to hear it. Nevertheless, the broadcast power was enormous: 1 million watts, twice as great as the largest commercial transmitter in history and directed with an antenna gain factor of tens of millions. Any

twin of Arecibo in the right part of M13, if it is tuned to the right frequency and pointing our way, won't be able to miss it when the signal reaches the cluster 22,000 years from now.

- ▶ The most interesting aspect of the Arecibo message is not the remote chance that it will be received but the fascinating strategy used to solve the anti-cryptography problem. The message relies on things we must share with the intended recipients, alien though they may be: mathematics and logic; the facts of physics, chemistry and astronomy; and the physical characteristics of the signal itself. Anyone in the Universe who can build a radio telescope must know these things. They are an information basis on which we can build a shared code.
- ▶ There have been many other attempts to send signals to extraterrestrials. Transmissions from other radio telescopes have been directed toward several stars within a few dozen light-years of Earth. Each has adopted its own solution to the anti-cryptography problem. The messages have used error-correcting codes, made attempts at logical or pictorial languages, and even played a short concert of electronic music.
- ▶ Communicating with aliens is, however, a fairly controversial subject.
 - Not every scientist believes it is wise to deliberately try to contact beings about whom we know nothing. Maybe we should listen first.
 - And not everyone thinks that radio signals are the most likely means for interstellar communication. Light signals offer many advantages, and in fact, there are several telescope-based projects underway to search for extraterrestrial light signals.
 - Some people also advocate more direct communication. The two Voyager spacecraft, heading out into interstellar space, each carry a gold-plated copper disk designed by Drake and Sagan, inscribed like a phonograph record with digital data that includes photographs of Earth and recorded greetings in more than 50 languages. Within its protective covering, one of those disks might retain its information for 100 million years.

- Other scientists argue that the best way to find and contact extraterrestrials would be to launch self-replicating probes, like von Neumann's reproducing robot. The von Neumann probes would travel to other solar systems and, using local materials, would assemble copies of themselves. The copies would then head out to more distant stars. The probes would search for intelligent life and report their findings back to Earth, carrying our messages with them.

- We might wonder whether these speculative exercises in anti-cryptography suggest a way to get at that thing that Claude Shannon left out of information theory: the concept of meaning. Remember, in Shannon's theory, *information* is just the distinction between possible messages, not any significance those messages might possess. It's an abstraction. Indeed, it was a brilliant abstraction, because it opened a world of possibilities and laid the groundwork for a technological and scientific revolution.
 - Shannon's abstract concept of information allows us to understand how different forms of information are interchangeable and can be represented in the common currency of bits. It tells us how to measure information by entropy and how to use codes for data compression and error correction. It tells us about noise, signal, and bandwidth; how to keep secrets; and how to penetrate the secrets of others. Shannon information even tells us how to bet on horse races.

 - But every kind of communication, every kind of computation, happens within a context that determines its meaning. Thus, the information stored in DNA exists within a biochemical apparatus that expresses its base sequences as a library of protein designs—as the basic operating system of a living organism. Viewed another way, the DNA message is a record of the long history of life on Earth. Either way, DNA is not simply abstract information. It is information about something.

- There is a hint, too, in the algorithmic information of Kolmogorov and Chaitin. The information content of a binary string is exactly the set of instructions necessary to create it in the context of the universal computer. The meaning of the program is determined by the computer within which it runs.
 - Ultimately, the context of every message and every program is the Universe itself. Laws of thermodynamics and quantum physics govern how information is acquired, stored, processed, and erased. Those principles set the rules, and some of those rules can take us to places that even Shannon did not foresee. The elementary quantum phenomenon of Wheeler and the holographic ideas of Bekenstein and t' Hooft intimate that information really lies in the relation between the part and the whole, between the quantum and the cosmos.
 - As our own information universe—the universe of humans and the machines we have made—grows exponentially more complex, we need to remember how that little universe is embedded in the far larger and richer network of nature—because the meaning of all of our light pulses and electrical signals, all those symbols on stone or bits of wood, lies not within them but around them, in the context of the Universe.
- If we ever do receive a signal from an extraterrestrial intelligence, it will be a profound challenge to all the sciences of information to somehow extract the meaning from that message. Yet it is likely that the most meaningful part will be the part that is easiest to read—that first layer of the message: “People made this.” Even that profound message would, in a sense, be just 1 bit: a yes instead of a no; a 1 instead of a 0.

TERMS

anti-cryptography: The problem, formulated by physicist Philip Morrison, of devising messages that can be understood even by a recipient who does not share a code with the sender, but who may share such things as mathematics, logic, the laws of nature, and the physical characteristics of the communication medium (e.g., the radio signal carrying the message).

Arecibo message: A signal sent once in 1974 as a demonstration of the powerful radar transmitter on the giant radio telescope in Arecibo, Puerto Rico, aimed at the globular cluster M13, more than 20,000 light-years distant. The message was designed by Frank Drake and Carl Sagan to be understandable by hypothetical extraterrestrial beings and includes binary arithmetic, the structure of DNA, numerical data about human beings and the human genome, a simplified diagram of the Solar System, and a diagram of the Arecibo radio telescope.

READINGS

Benford, *Deep Time*, parts I and II.

Sagan, ed., *Communication with Extraterrestrial Intelligence*.

QUESTIONS

- 1 In the 1957 science fiction story “Omnilingual” by H. Beam Piper, space archaeologists excavating the ruins of an ancient Martian civilization are unable to read the Martian inscriptions until they discover their Rosetta Stone: a periodic table of the elements. Explain. Are there real examples of using shared scientific knowledge to interpret ancient inscriptions on Earth?

- 2** Another science fiction story, the 1961 British TV serial *A for Andromeda* (written by noted astrophysicist Fred Hoyle), involved an alien invasion from a distant galaxy. This invasion, however, is conducted entirely by radio. How is this possible?

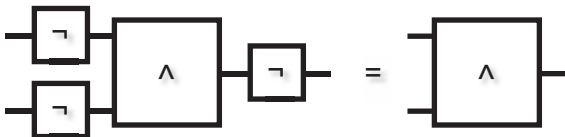
Answers to Questions

Lecture 1

- 1 Unicode has $2^{16} = 65,536$ possible characters.
- 2 My own computer contains text files—both in ASCII and several other formats—image files of at least six different types, numerous sound and video clips, many programs that I have written myself, numerical data files, spreadsheets, presentation slides, databases, compressed archives, 3-D modeling files, stored web pages and e-mails, and several files that I cannot immediately identify. Your answer may vary!
- 3 *Information* is actually a remarkably old word, dating back more than 500 years. It comes from a Latin word *informatio*, meaning “concept” or “idea”—a version of the verb *informare*, meaning “to shape” or “to form.” This is not far from the modern usage as a term for that which is communicated.

Lecture 2

- 1 Babbage’s difference engine was motivated in part by the difficulty of constructing reliable mathematical tables. Kulik’s problem was woefully familiar to him. Thus, he would have certainly understood this application of Shannon’s electronic computer.
- 2 In logic terms, $a \vee b = \neg[(\neg a) \wedge (\neg b)]$. This is easily verified in a logic table. We can diagram this as follows:



Lecture 3

- 1 The following table is useful in answering this question:

# of lights	# of signals	$H = \log_2(M)$	bits/codeword	bits/light
1	$3^1 = 3$	1.585 bits	2 bits	2 bits/light
2	$3^2 = 9$	3.170 bits	4 bits	2 bits/light
3	$3^3 = 27$	4.755 bits	5 bits	1.67 bits/light
4	$3^4 = 81$	6.340 bits	7 bits	1.75 bits/light
5	$3^5 = 243$	7.925 bits	8 bits	1.60 bits/light

Note: To calculate $\log_2(x)$, find $\ln(x)$ and divide by $\ln(2) = 0.6931$. Thus,

$$\log_2(3) = \frac{\ln(3)}{\ln(2)} = \frac{1.0986}{0.6931} = 1.585.$$

- 2 Each letter has entropy of $\log_2 26 = 4.70$ bits, and each digit has $\log_2 10 = 3.32$ bits. The total amount of car-identification entropy is, thus: $(3 \times 4.70) + (4 \times 3.32) = 27.38$ bits. This is enough to identify about 175 million different cars; as it happens, Ohio has only about 12 million vehicles registered statewide.
- 3 Bob's initial missing information is $\log_2(100) = 6.64$ bits. After he learns whether the number is even, there will (in either case) be 50 possibilities, and $\log_2(50) = 5.64$ bits. He learned exactly 1 bit! He will need six more questions (just more than 5.64) to determine the number.
- 4 First, ask: Is the word in the first half of the dictionary? Next, ask which of the two possible quarters of the dictionary the word is in, and so on. At any stage, you have a range of possible pages in the dictionary; the key idea is to divide that page range in half and find out which half is correct. (Eventually, you narrow things down to a single page. At that point, you ask whether the word is on the top half of the page, and so on.)

Lecture 4

- 1 The answer depends on the table you consult, but for one online table, the answer was 11—the letters Y, W, G, P, B, V, K, X, Q, J, and Z combined have about the same frequency as E. (Because the table I consulted did not include spaces, this was about 12%.) The total probability for the 12 most common letters was about 80%.
- 2 The surprises (in bits) are

		<i>T</i>	
		<i>p</i>	
<i>S</i>		warm	cool
	sunny	1.32	2.32
	cloudy	3.32	1.74

- Warm and cloudy, the least likely type of weather, is the most surprising. The entropies are given by the average surprises: $H(S) = 0.97$ bits, $H(T) = 1.00$ bit, and $H(S, T) = 1.85$ bits (the weighted average of the surprises in this table).
- 3 Each person has a probability of 50/300 million of being struck in the next year, and the probability of being struck tomorrow is 1/365 of that. Thus,

$$p = \frac{50}{300,000,000 \times 365} = 9 \times 10^{-12},$$

less than one chance in 100 billion. The surprise $s = \log_2(1/p) = 36.7$ bits. When we flip a coin many times, the surprise increases by 1.0 bit for each heads; thus, getting 40 heads in a row has a surprise of 40 bits, even more surprising than getting struck by lightning.

Lecture 5

- 1 My most commonly used abbreviation is *BTW* for “by the way.” I also use *PB* for “peanut butter” and *NG* for “no good.”
- 2 None of the codewords is a prefix of other codewords. The Kraft sum is $2^{-2} + 2^{-3} + 2^{-1} = 7/8$, which is less than 1. Here are some common letters and their codewords:

E 100 (dot-space)
T 1100 (dash-space)
A 101100 (dot-dash-space)
O 1101101100 (dash-dash-dash-space)
I 10100 (dot-dot-space)
N 110100 (dash-dot-space)

Notice that every letter must end with a 0 to give an extra-long pause between letters. Otherwise, A (dot-dash) would be the same as E followed by T.

- 3 If you have access to a data-compression program, this is worth trying. After the first compression, further applications of the program do not compress much further; indeed, they may actually begin to expand the file slightly, as happened with the random file. The data compression does not continue to shrink the file more and more. You cannot compress a novel to a few bytes!

Lecture 6

- 1 No computer enhancement can provide information that does not exist in the original image. If the blurry original does not contain the “high-frequency” Fourier data, it cannot be restored. (Some enhancement is sometimes possible to bring out information that is actually in the original image, but the movies take this way too far!)

- 2 A harp is much like a piano, minus the keys, the case, and the dampers. Another slightly more far-fetched example would be a pipe organ. Each pipe corresponds to a sound frequency, and a loud external sound might set up corresponding “echoes” within them. In each case, the musical instrument includes many natural “vibrators” (strings or air pipes) with a wide range of frequencies. An instrument with only one or a very few vibrators (a flute or a violin) would not make a good Fourier transform.

Lecture 7

- 1 Bob’s information about what Alice sent is $H(X) - H(X|Y)$. Alice’s information about what Bob receives is $H(Y) - H(Y|X)$. Both of these equal the mutual information $I(X;Y)$, the common “overlap” in the Venn diagram of information quantities.
- 2 For this error rate, the binary entropy function $h(0.001) = 0.0114$ bits. This is the information loss per use of the channel. In a 1000-bit message, we expect to lose about 11.4 bits. (This makes sense. Even with a single error, it would take about $\log_2(1000) = 9.97$ bits of “correction data” to locate it—and there may sometimes be more than one error.)

Lecture 8

- 1 Amusingly, the word *alooof* is one such.
- 2 The number of codewords is bounded by

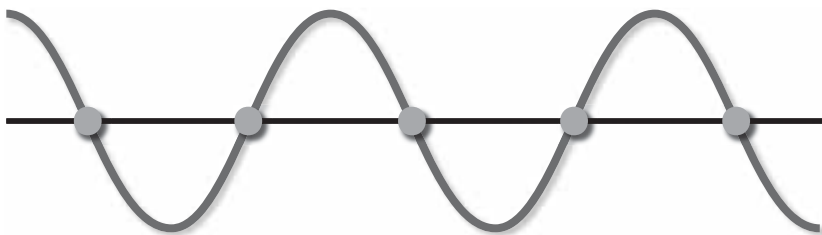
$$\frac{2^n}{n+1} = \frac{2^7}{8} = 16,$$

which happens to be the number of codewords when $k = 4$. Thus, the $(7,4)$ code just satisfies the bound. (Not all codes work out so perfectly, alas, and the bound for $d > 3$ is not as simple.)

- 3** The answer is twofold. First, because there might not be any error at all, there are five possible “error syndromes” (as the jargon has it). We need enough error-correcting information to tell which one we have. Furthermore, when we use additional bits, they might also introduce errors, which we must be able to correct. The three error-correction bits for the (7,4) code can recognize $2^3 = 8$ syndromes—no error plus seven possible 1-bit errors.

Lecture 9

- 1** It is simplest to look at a picture. Time runs along the horizontal line below, and every dot is a sample point. The signal happens to be zero at every sample. Because the dots occur 200 times per second, the wave frequency is half as great: 100 Hz.



This is why we need at least $2W$ samples per second to characterize a signal of bandwidth W . With too few samples, they can be “fooled” by a high-frequency wave.

- 2** If we double W , we always double \mathcal{C} . Thus, the question is whether doubling P/N can ever increase $\log_2(1 + P/N)$ by that much. Some calculator experiments show that, if P/N is very small, doubling it *almost* doubles the logarithm. But for large P/N , the logarithm does not increase by very much. Thus, it is always better to double W , if you can leave P/N the same.

Lecture 10

- 1 The Wikipedia page on ROT 13 lists several examples. My favorites are *irk-vex* and *terra-green*, which have an interesting coincidence of meaning, and *gnat-tang*, which simply reverses the input word.
- 2 Many people use a special private system to generate new computer or website passwords. Because someone who knows the generating system would be able to guess the passwords easily, the security relies on the secrecy of the method. Given that there are many possible methods, this approach may be secure enough for casual usage. But its security is somewhat fragile. If one generated password is compromised somehow, an adversary may be able to guess all the others.

Lecture 11

- 1 Both are letter-exchange ciphers. That is, the cipher system simply swaps pairs of letters. This means that the encoding and decoding processes for a message are exactly the same.
- 2 A random 20-letter key phrase has $H(K) = 20 \times 4.7 \text{ bits} = 94 \text{ bits}$. With $D = 3.2 \text{ bits/letter}$ for English, this yields: $L = H(K)/D$ 29 letters. This seems unreasonably short—less than 50% longer than the code phrase itself! This is certainly not enough for the Babbage method.
- 3 Yes! Because cryptanalysis relies on redundancy and some data compression could remove redundancy in the plaintext, this is an excellent approach.

Lecture 12

- 1 It might happen that some common plaintext word or phrase occurs in the same location in two messages coded with the same key. This would lead to exactly the same series of letters in the ciphertext. Thus, searching for such “coincidences” in the ciphertexts would be one way to search for reuse of the key.
- 2 The computer stores only an encrypted form of each password. When a user types in the password, the computer encrypts it and compares the result to the encrypted password on file. No record is kept of the user’s input, and no one possesses a decryption key. Thus, the list of encrypted passwords is not enough to give anyone access to the user accounts. (Something like this is actually standard practice on all servers.)

Lecture 13

- 1 Given that there are about 6×10^9 base pairs in human DNA, the total length is $(6 \times 10^9) \times (0.33 \times 10^{-9} \text{ m}) = 2 \text{ m}$. My DNA is slightly longer than I am.
- 2 One base pair means just two possibilities: AT and TA. With four bases per codon, this gives $2^4 = 16$ possible codons. If the genetic code is like the one on Earth, however, at least one of these must be a stop codon, leading to a maximum of 15 amino acids. However, this would mean that each amino acid had only one possible codon, which means that every mutation would change the corresponding protein. The code would be much less fault-tolerant than ours.

Lecture 14

- 1 In fact, natural selection corrects no errors in any single molecule. Instead, it simply makes it less likely that errors reproduce. The error correction applies only to a whole population of replicating molecules.

- 2 We know that RNA can act as both an information-bearing molecule and an enzyme. DNA carries information but is a poor enzyme (because its functional parts are protected on the inside of the double helix). Proteins are enzymes, but they seem to be informational dead ends; we see no examples of the reproduction of protein information in the basic biochemical machinery of life.
- 3 In many ways, prion diseases are more analogous to the inorganic tin disease. The original misshapen protein can arise by accident, without any parent.

Lecture 15

- 1 One petabyte is 1000 terabytes; thus, the 2.5-petabyte system would cost \$125,000 and weigh more than 1000 pounds. Moore's law (see Lecture 1) suggests that these numbers will decline very rapidly. (The trick, of course, is figuring out how to read and transfer the brain's information!)
- 2 A vocabulary of 256 words means an entropy of 8 bits per word, yielding $3.8 \times 8 = 30.4$ bits/second. A 5000-word vocabulary means 12.3 bits per word, yielding $2.7 \times 12.3 = 33.2$ bits/second. The capacity is, thus, around 30 to 33 bits/second.

Lecture 16

- 1 As a rough guess, I believe my entire knowledge of the gas, including all relevant information about its history and present state, is much less than 1 gigabyte, around 10^{10} bits. In the lecture, we estimated the missing microstate information at 6×10^{23} bits, or more than 60 trillion times as much. Comparatively speaking, I know almost nothing about the gas in the jar!

- 2** The change in thermodynamic entropy is

$$S = \frac{Q}{T} = \frac{2000}{373} = 5.4 \text{ joules/kelvin.}$$

This corresponds to a Shannon information loss $\Delta H = \Delta S/k_2$, which is around 5.4×10^{23} bits. This corresponds to around 16 bits/molecule.

- 3** It is precisely the principle of microstate information that requires that a macrostate cannot evolve into a “smaller” macrostate; that is, Texas cannot squeeze itself into Delaware. Far from being incompatible, one is the necessary basis for the other.

Lecture 17

- 1** One idea would simply be to slow the computer down. In our simple capacitor example, diminishing the speed by a factor of 10 reduced the power output by 100 times and the total dissipation of a computation by 10 times.
- 2** The performance of such devices as the Ivy Bridge processor is already limited by waste heat. Making the devices denser and faster without a corresponding reduction in waste heat would simply melt the devices!
- 3** If the gate has n input bits, then it has 2^n possible input states. It needs to have at least as many output states, because each different input leads to a different output. Therefore, it needs at least n output bits. But a reversible gate can be run backward, exchanging input for output—and that means that input and output bit numbers must be exactly equal.

Lecture 18

- 1** The entropy H is the entropy of a 90% / 10% probability distribution, which is 0.47 bits. The tip thus contains $I = 1 - H = 0.53$ bits. We should bet 90% of our money on the more likely horse and 10% on the less

likely horse. The gain factor is $G = 2' = 1.44$. We'd have to race about 19 times, because $144^{19} \approx 1000$. However, in this set of races, we expect to lose about twice. Each time that happens, we lose 80% of our wealth (because we are paid back with decimal odds 2.0 for a bet of 10% of our wealth).

- 2 Your answers may vary. Because I am fairly risk-averse financially, Samuelson's arguments make sense to me. I am also aware that the "long run" may be too long a timescale for planning my own finances. Nevertheless, the concept of hedging and the exploitation of exponential growth are both good ideas. I would be tempted by a more conservative partial-Kelly strategy.

Lecture 19

- 1 To make my laptop truly universal, we must supplement it with an unlimited amount of external storage—an endless supply of memory sticks, for instance, that can be plugged in and swapped out at will. Then, the laptop and the Turing machine will be fundamentally equivalent.
- 2 Shannon's first fundamental theorem (the data compression theorem) says that we can find a code for the output of the source that uses only about H bits. We could use such a code in a UM program for describing a particular string. This program would thus have not much more than H bits. A few particular strings might have shorter programs, but that would not affect the average much. Thus, $e[K(s)] \approx H$.
- 3 In ordinary probability, *random* refers to the process by which new events are generated. The process is random if it cannot be predicted. The process is random, not the individual events. But algorithmic information theory defines an individual sequence as random provided its algorithmic entropy, $K(s)$, is about equal to its length, $L(s)$. Such a string is "incompressible," in that we cannot provide a shorter description of it. Note the failure of the gzip data compression program to make our random file shorter back in Lecture 5.

Lecture 20

- 1 One interesting example would be the gzip data compression program from Lecture 5. I could apply the program to the program file itself. I tried this experiment. The original gzip program was 92,600 bytes long; the compressed version (the output of `gzip,gzip`), so to speak) was 45,300 bytes long.
- 2 The key question is what constitutes a description of a number. Some phrases are descriptions of numbers; others are not. (The non-description phrases are like the non-halting programs on the UM; they have no definite output.) However, we have no universal and infallible way to decide exactly which phrases are true descriptions. Thus, Berry's phrase is not exactly a definite description of a number.

Lecture 21

- 1 Probably the best response would be to ask Physicists A and B to define their terms and explain their positions more clearly. Each one faces some difficulties. For instance, the probabilities associated with an individual photon are still continuous, while the superposition states are not really distinct from one another in the same sense that 0 and 1 are. Quantum physics is not just a “more analog” or “more digital” version of classical physics; it is actually something quite new, with new rules.
- 2 Decoherence depends on the formation of physical “records” in the environment of some property of an object. Large objects interact more readily with more parts of the environment—atoms, photons, and so on—thus, this information exchange happens more rapidly. It is much easier for a small system to be informationally isolated for a significant amount of time.
- 3 At this stage, very soon before 2019, it is probably wisest to take the negative side of the bet, that is, against my position. On the other hand, given the present rate of progress, 20 more years might make a considerable difference!

Lecture 22

- 1 No. In Shannon's theory, no kind of correlation between two systems prevents a third system from sharing the correlation.
- 2 The no-signaling principle, satisfied by quantum entanglement, forbids this.
- 3 With C and E, they can create K—that's quantum cryptography. But E and K are not enough to create C (by the no-signaling principle), and C and K cannot create E (because this would just be an LOCC operation, which cannot create entanglement).

Lecture 23

- 1 The cube's area is 6×10^{-2} square meters, which yields 10^{-25} square meters per bit. That's pretty small, but it is still 1.44×10^{44} times larger than the fundamental limit!
- 2 Your answers may vary. As for myself, I think that the search for good information axioms can help us to understand the deeper structure of quantum theory, even if we do not take any particular set of axioms too seriously. (After all, someday, we may have a better theory than quantum mechanics!)
- 3 Again, you will have your own answer. I am particularly fascinated by the relational view of information that seems to be at the root of the holographic principle.

Lecture 24

- 1 Because the periodic table expresses universal laws of chemistry, the Martian periodic table will have a recognizable shape. The other symbols can then be interpreted based on their position within the table—hydrogen, oxygen, and so on. Ancient Earth civilizations did not

know about the periodic table, but they did have calendars. Knowing astronomical data about the Sun, Moon, and planets has helped to decipher Mayan hieroglyphs, to take one example.

- 2** The plot of *A for Andromeda* is a brilliant exercise in the transformability of information. The alien radio signal includes plans for a computer and the genetic code of an artificial alien organism. Human scientists, not realizing the danger, carry out the radio instructions and initialize the invasion.

Key Equations in the Science of Information

Entropy (Lectures 3 and 4)

How much information is in a message?

The entropy H of an information source is the fundamental measure of the amount of information produced by the source. There are two equations for the entropy H of a message source in bits. The first is the Hartley-Shannon entropy:

$$H = \log_2(M),$$

where M is the number of possible messages from the source, each message considered to be equally likely. The Shannon entropy generalizes this for a source whose messages are not equally likely:

$$H = \sum_x p(x) \log_2 \left(\frac{1}{p(x)} \right),$$

where $p(x)$ is the probability of message x . The logarithm

$$\log_2 \left(\frac{1}{p(x)} \right)$$

is called the surprise of message x . Shannon entropy is average surprise.

According to Shannon's first fundamental theorem, the entropy H equals the minimum average number of bits necessary to code the messages produced by the source. That is, H tells us how well we can accomplish data compression.

Information Inequality (Lecture 5)

Knowing the probabilities means less surprise, on average.

This somewhat odd-looking relation expresses a basic property of the Shannon entropy. If $p(x)$ is the actual probability distribution for messages and $q(x)$ is some other conceivable distribution, then:

$$H \leq \sum_x p(x) \log_2 \left(\frac{1}{q(x)} \right).$$

Equality holds only when the probabilities are equal: $p(x) = q(x)$.

The intuitive way to understand this relation is to imagine two people describing the same information source. Fred knows the “correct” probabilities $p(x)$ for the messages, while George mistakenly thinks the probabilities are $q(x)$. Different probabilities lead to different measures of the “surprise” of the messages. Sometimes Fred is more surprised than George, and sometimes the reverse is true, but on average, George’s surprise is greater.

We used the information inequality to show that no prefix-free code can have an average codeword length less than the Shannon entropy (Lecture 5) and to show that the best way to hedge horse race bets is to make our “betting fraction” equal to the probability of winning for each horse in the race (Lecture 18).

Kraft Inequality (Lecture 5)

Not all codewords can be short.

A binary code can have codewords of different lengths. However, in order to be able to divide a long string of codewords into separate messages, we need to use a prefix-free code, one in which no codeword is an initial segment of another codeword. This requirement imposes a requirement on the codeword lengths.

Consider a set of binary codewords for the messages x from a source. The codeword for x has length $L(x)$ bits. Then, the code can be prefix-free only if the following is true:

$$\sum_x 2^{-L(x)} \leq 1.$$

We proved this by constructing a binary tree of codewords and imagining a unit amount of water flowing along its “pipes” and dividing equally at each branch.

We can add two additional comments. First, if we fix a set of lengths satisfying this inequality, we can always find a prefix-free code with codewords of these lengths. Second, we can always trim our codewords (as in Huffman coding) so that the inequality becomes an equation, that is, the left-hand side equals 1.

In Lecture 19, we saw the Kraft inequality again as a mathematical condition on the lengths of the possible programs on our universal computing machine.

Mutual Information (Lecture 7)

How much information gets through the noise?

When a communication channel has noise, information is lost in transmission. The mutual information I measures the amount of information conveyed in a noisy channel. If X is the input message and Y is the channel output, then:

$$I(X; Y) = H(X) - H(X | Y) = H(X) + H(Y) - H(X, Y).$$

Here, $H(X|Y)$ is the entropy of the input given the output, and $H(X,Y)$ is the joint entropy of the input-output pair.

This has a simple interpretation: $H(X)$ is the amount of information that the receiver lacks about the message before any communication takes place. Once the channel output Y arrives, the receiver still lacks information $H(X|Y)$. The amount of information gained by the receiver is, therefore, $I(X;Y)$.

By Shannon's second fundamental theorem, the capacity C of a channel is just the maximum of the mutual information $I(X;Y)$, considering all possible inputs. We can send any amount of information up to C through the channel, using error-correcting codes to repair all the errors that the channel might produce.

Though many powerful error-correcting codes are now known, almost none were discovered before Shannon proved his great theorem. Knowing a thing is possible is a crucial step to figuring out how to do it!

Here is an example: For the binary symmetric channel, in which 0 and 1 each have an error probability e , the maximum mutual information (channel capacity) has an especially simple form: $C = 1 - h(e)$, where $h(e)$ is the binary information function, the Shannon entropy of a binary source with probabilities e and $1 - e$. If $e = 0.1$ (a 10% error probability), then $h(e) = 0.469$ bits, so the capacity is just $C = 1 - h(e) = 0.531$ bits per use of the channel. Any transmission rate less than this is possible, with negligible overall likelihood of error in the long run.

Capacity of an Analog Signal (Lecture 9)

How many bits can we send via an analog signal?

When we send bits of digital information through an analog channel (radio transmission, electrical signal in a wire, sound waves in air), then the rate at which we can reliably send information is limited both by the bandwidth of the channel and its signal-to-noise ratio (SNR).

The information capacity \mathcal{C} of an analog signal (e.g., a radio signal) in bits per second is:

$$\mathcal{C} = W \log_2 \left(1 + \frac{P}{N} \right),$$

where W is the bandwidth (in Hertz), and P/N is the ratio of the signal power (P) to the noise power (N). Note that N usually depends on W . (A channel with a wider bandwidth admits more noise.)

As an illustration, consider a radio signal with a bandwidth of 10 kHz and an SNR of 100. (This is often specified using the logarithmic decibel scale; 100 corresponds to 20 decibels.) The capacity is $(10 \text{ kHz}) \log_2(1 + 100) = 66,600$ bits per second. However, if external interference reduces the SNR to only 10 (10 decibels), then the capacity is reduced to 34,600 bits per second.

Unicity Distance (Lecture 11)

When is cryptanalysis possible?

The unicity distance L is Shannon's estimate of the amount of ciphertext that is needed to uniquely determine the plaintext of an enciphered message. If we have a sample of ciphertext longer than this, then we can in principle decipher the message. However, this does not take into account the computational difficulty of the code-breaking process.

If the key has entropy $H(K)$, then:

$$L = \frac{H(K)}{D},$$

where D is the redundancy of the plaintext per symbol. For English text, Shannon estimated that $D \approx 3.2$ bits/letter.

For instance, a simple substitution cipher has a key entropy $H(K) = \log_2(26!) \approx 88$ bits, giving a unicity distance of about 28 letters. If we have a sample of ciphertext much longer than this, and we know in advance that a simple substitution cipher is used, then we should be able to determine the original plaintext message.

Error Catastrophe (Lecture 14)

How much genetic information can survive in the long run?

In the earliest days of life on Earth, the simplest self-replicating molecules made copies of themselves in a very noisy environment. This noise limited the size of the molecular sequence that could preserve its information into the future.

Eigen showed that a self-replicating molecular sequence of length L will survive in spite of a copying error rate e , provided:

$$Le \leq \log(S) \approx 1,$$

where S is the survival advantage of the correct sequence compared to its imperfect copies. The estimate $\log(S) \approx 1$ is a heuristic value; we do not expect $\log(S)$ to be much larger than 1 in the conditions of the earliest self-replicating molecules.

As a modern-day illustration, viruses replicate their DNA with an error rate of around 10^{-4} ; because their DNA genome is only about 10^4 bases long, they just avoid the error catastrophe.

Thermodynamic Entropy (Lecture 16)

What is the link between bits and heat?

The concept of entropy came from thermodynamics in the 19th century. Boltzmann discovered the relation between thermodynamic entropy and information entropy, though it was many decades before Shannon's information theory made the meaning of his formula plain. The thermodynamic entropy S is given by:

$$S = k_2 \log_2(M) = k_2 H,$$

where k_2 is approximately 10^{-23} joules/kelvin per bit and M is the number of microstates consistent with our macroscopic data. Thus, H is the information in bits that we lack about the detailed microstate of the system. Because the constant k_2 is so tiny, a huge amount of microscopic information corresponds to a comparatively tiny amount of thermodynamic entropy.

Entropy had been discovered earlier by Clausius, and we can determine how it changes from the macroscopic processes a system undergoes. If heat Q is added to a system of absolute temperature T (in kelvins), then the entropy changes by $\Delta S = Q/T$.

Landauer's Principle (Lecture 17)

Erasing information has an inescapable cost.

Landauer studied the basic physics of computer operations, including those that erase information. He found that there is an inescapable thermodynamic cost for erasing data. If information is erased, the environment must increase its entropy by at least $\Delta S \geq k_2$ per bit. If the environment has absolute temperature T , then this corresponds to a waste heat of at least $k_2 T$ per bit.

Of course, k_2 is very small (10^{-23} joules/kelvin per bit). Actual existing computers produce far more waste heat than this in their operation. But Landauer's principle sets the absolute minimum amount of waste heat that any computer will have to produce.

It is Landauer's cost of information erasure that keeps Maxwell's demon from violating the second law of thermodynamics. Acquiring microstate information does permit the demon to reduce the entropy of a system, but that entropy reduction is offset when the demon "clears" its memory of useless data later on.

Kelly's Gain Equation (Lecture 18)

Each bit of extra information doubles your gambling gain.

Kelly introduced a new way of looking at information. Shannon said that information is the ability to distinguish between alternative messages. According to Kelly, information is the betting advantage that the message can provide in a gambling game.

Suppose we are betting on a horse race with ideal equal odds, and we distribute our bets among the possible results in the best possible way (that is, betting a fraction $p(x)$ of our wealth on each outcome x). Then a tip containing information I about the race result will allow us to achieve a gain factor per race of G , given by: $\log_2(G) = I$ (in exponential form, $G = 2^I$).

With no tip, $I = 0$ and, thus, $G = 1$. We do not expect either to win or lose money in the long run. However, if the tip tells us the result of a binary horse race, then $I = 1$ bit and $G = 2$. We expect to double our money per race. Less reliable or informative tips provide a smaller advantage.

This G is the best expected gain factor in the long run. But anyone who actually bets by Kelly's log-optimal strategy should expect some dramatic ups and downs in wealth along the way.

Algorithmic Entropy (Lecture 19)

Entropy equals the length of the minimal description.

The algorithmic entropy (also known as the Kolmogorov complexity) represents an entirely new approach to information, one based on computation rather than communication. Essentially, the algorithmic entropy of s is the size of the smallest description of s .

Given a particular universal computing machine (UM), s^* is the shortest program that produces the string s as output and then halts, that is, the shortest program in $P(s)$. Then, the algorithmic entropy of s is:

$$K(s) = L(s^*),$$

where $L(x)$ is the length of string x in bits. If s has a pattern, then $K(s) < L(s)$. If s is random, then $K(s) \approx L(s)$.

Zurek suggested that the thermodynamic entropy of any system that includes a computer should be the sum of two parts: the regular Boltzmann entropy, $k_2 \log_2(M)$, and a new algorithmic term, $k_2 K(m)$, taking into account the particular memory bits m . However, because of the Berry paradox, the algorithmic information is an uncomputable function, like Turing's halting function and the halting probability Ω .

“Monkey” Probability (Lecture 19)

How likely is it for a random program to output s ?

The concept of monkey probability lets us translate algorithmic information ideas into a kind of probability. If we allow a monkey to randomly type a program into our UM, then the likelihood that it types in a program that produces s as output is:

$$\text{Prob}(s) = \sum_{p \in P(s)} 2^{-L(p)} \approx 2^{-L(s^*)} = 2^{-K(s)}.$$

The monkey probability for s is dominated by the shortest program that produces s , the program denoted by s^* .

This provides an absolute sense in which some strings s are “more likely” than others. Ray Solomonoff used this idea to give a new perspective on Ockham’s razor, the logical principle that simpler theories are more likely theories. More likely in what way? In monkey probability!

Monkey probability is uncomputable. In fact, the sum of the monkey probabilities for all finite output strings s equals Ω , Chaitin’s uncomputable halting probability.

Black Hole Entropy (Lecture 23)

What is the area of 1 bit of information?

According to Bekenstein, the entropy of a black hole is:

$$S_{\text{black hole}} = k_2 \log_2(M) = \text{constant} \times A_{\text{horizon}},$$

where M is the total number of histories that could have led to the present black hole, and thus, $\log_2(M)$ is the number of bits that have “vanished” into the hole. A_{horizon} is the area of the black hole event horizon. As discovered by Hawking, the constant above is such that 1 bit of missing information corresponds to an area of about $7 \times 10^{-70} \text{ m}^2$.

A black hole the mass of our Sun has a radius of about 3 km, yielding a horizon surface area of more than 100 million square meters. Such a black hole would correspond to a truly vast missing information of more than 10^{77} bits.

The question of whether this information is truly lost or whether it will later reemerge via quantum radiation from the black hole, is called the *black hole information problem*.

The same connection between information and area motivates the holographic principle, according to which we can suppose that the information content of any region of space is really located on its outer boundary, with each bit occupying no more than $7 \times 10^{-70} \text{ m}^2$.

Glossary

algorithmic entropy: The length of the shortest program on a universal computer that prints a specified string as output and then halts. Formulated by Andrey Kolmogorov and Gregory Chaitin as the basis for algorithmic information theory, in which the information content of any object is simply the length of its shortest description on a computer. The algorithmic entropy of a binary string is an uncomputable function, thanks to the **Berry paradox**.

analytical engine: The general-purpose mechanical computer whose design was sketched, but never completed, by Charles Babbage in the 19th century, anticipating most of the key ideas of modern computers. The computer could be programmed using punched cards similar to those that controlled the Jacquard loom, with a “mill” for performing computations and a “store” for working memory.

anti-cryptography: The problem, formulated by physicist Philip Morrison, of devising messages that can be understood even by a recipient who does not share a code with the sender, but who may share such things as mathematics, logic, the laws of nature, and the physical characteristics of the communication medium (e.g., the radio signal carrying the message).

Arecibo message: A signal sent once in 1974 as a demonstration of the powerful radar transmitter on the giant radio telescope in Arecibo, Puerto Rico, aimed at the globular cluster M13, more than 20,000 light-years distant. The message was designed by Frank Drake and Carl Sagan to be understandable by hypothetical extraterrestrial beings and includes binary arithmetic, the structure of DNA, numerical data about human beings and the human genome, a simplified diagram of the Solar System, and a diagram of the Arecibo radio telescope.

ASCII: The American Standard Code for Information Interchange, a widely used 7-bit code for text data first introduced in the early 1960s to represent any key on a standard English keyboard, including numerals, punctuation marks, and both uppercase and lowercase letters. Because most computers store ASCII symbols as bytes of 8 bits, the first bit is always a 0.

bandwidth: The range of frequencies used by an analog signal, be it sound, radio, or an electrical voltage. Every communication channel has a limited bandwidth, a finite range of frequencies that can be conveyed faithfully. Along with the **signal-to-noise ratio**, the bandwidth limits the information rate of an analog signal, according to the Shannon-Hartley formula.

Baudot code: Teletype code first introduced by Émile Baudot in 1870 and later much revised for international use, with each letter represented by a codeword of 5 bits and additional symbols (such as numerals) expressed by the use of a special codeword that shifts to a second, alternative set of codewords. For data storage, the 1s and 0s can be represented by small holes punched in a paper tape.

Berry paradox: The paradox contained in the following question: What is the smallest integer that cannot be described in 12 words or less? Given that some such numbers exist, there must be a smaller one. However, this 12-word phrase itself appears to describe that number, so it is describable in 12 words after all. A computational version of this paradox shows that the algorithmic entropy is uncomputable in general.

binary erasure channel: A simple communication channel with two possible inputs (0 and 1) and three possible outputs (0, 1, and B, for “blank”). The only possible errors replace the input bit with B. This is analogous to a student filling out a true-false test but sometimes leaving an answer blank.

binary symmetric channel: A simple communication channel with two possible inputs and two possible outputs, both of which may be designated 0 and 1. The channel is “symmetric” because the likelihood of an error (changing 0 to 1 or 1 to 0) is the same whether the input is 0 or 1.

bit: The fundamental “atom” of information, the binary distinction between two alternatives (1/0, yes/no, on/off). Any type of information can be expressed as a sequence of bits.

black hole: A massive object whose extreme gravitation prevents even light from escaping. The hole is surrounded by a mathematically defined surface called its *event horizon*; no information from within the horizon can

get outside. However, thanks to quantum mechanics, black holes are not entirely black: They emit a very faint thermal radiation. Black holes have an entropy proportional to their horizon area. This entropy represents the information that has been “swallowed” by the black hole.

black hole information problem: Does information that falls into a black hole ultimately disappear from the Universe, or does it eventually reemerge in the quantum radiation from the hole? The subject of a famous bet, in which physicist John Preskill, later joined by Stephen Hawking, argued that the information eventually returns. Kip Thorne was reportedly not yet convinced.

Boolean logic: The algebraic logic devised by George Boole in the 19th century, used as the basis for the design of fully electronic computers in Claude Shannon’s 1937 master’s thesis. Shannon showed that any mathematical calculation could be reduced to a complex network of Boolean operations and that these could be implemented via electrical circuits.

Braille code: The first binary code, introduced as a system of writing for blind readers by Louis Braille in 1837. Each letter is represented by a codeword of 6 bits, arranged as a rectangular pattern of raised dots on paper. Additional symbols are available by the use of special “shift” symbols.

byte: A group of 8 bits in a computer memory, generally the smallest block of information that can be addressed individually.

channel capacity: The maximum mutual information for a communication channel. According to Claude Shannon’s second fundamental theorem, the channel can be used to send any number of bits less than the capacity, with a negligible overall probability of error. (This theorem thus guarantees the existence of effective and efficient error-correcting codes, though it does not show how to construct them.)

ciphertext: In a cipher system, the coded message that is actually transmitted and could be intercepted by an eavesdropper. Without the key, however, the plaintext remains secret.

codon: A group of three successive bases in DNA, forming a codeword that represents a single amino acid in a protein. With four base pairs (AT, TA, CG, and GC), there are 64 possible codons, which is more than enough to represent the 20 possible amino acids. (There are also “start” and “stop” codons to designate the ends of the protein sequences.)

conditional entropy: The average entropy based on conditional probability. Thus, in a noisy channel, the conditional entropy of the input message given the channel output is a measure of the information that the receiver still lacks about the message even when the output is known. Conditional entropy is, therefore, a measure of the noise in a channel.

conditional probability: The likelihood of an event given that some other event also takes place.

cryptanalysis: The art and science of penetrating other people’s secret codes, deducing their plaintext messages from intercepted ciphertexts. Shannon’s information-theoretic approach to cryptography showed when cryptanalysis is possible in principle, but the practicality of the task often depends more on its computational difficulty.

decibel: A logarithmic measure of the ratio of two magnitudes, often used to describe the signal-to-noise ratio (SNR). Each decibel difference of 10 represents a factor of 10 in power. Thus, if signal A is 30 decibels more than signal B, it has 1000 times the power. The decibel scale is often used to measure the loudness of a sound relative to the quietest sound that a normal human being can hear.

decimal odds: A way of stating the betting odds of a horserace or other gambling game. If a particular horse has decimal odds of 5.0, this means that a bet that costs \$1 will pay winnings of \$5 if the horse wins. (The same odds are sometimes called “4 to 1” odds, using the old British system.) In perfectly fair, ideal betting odds, the horse would have a 1/5 probability of winning.

delayed-choice experiment: A type of quantum two-slit experiment on a photon, devised by John Wheeler. We can either obtain “which slit?” information or observe interference effects but not both in the same

experiment. In the delayed-choice version of the experiment, the decision about which experiment to perform can be put off until long after the photon has passed slits. Thus, the elementary quantum phenomenon that the experiment represents is not completely localizable in space and time.

difference engine: The first mechanical computer devised by Charles Babbage in order to create and print mathematical tables without human error.

differential analyzer: One of several related analog computers of the early 20th century. Each used special gear systems to connect the rotations of different shafts, the mechanical movement of each shaft representing a continuously changing variable, together representing the solution to a system of differential equations, such as the equations governing ballistic motion.

DNA: Deoxyribonucleic acid, the famous double-helix molecule that is the basic genetic information storage system for Earth life.

eidostate: A neologism (based on the Greek word meaning “to see”) for the state of a thermodynamic system, including all of the information available to Maxwell’s demon, whether it is macroscopic or microscopic.

Enigma system: Sophisticated encryption system used throughout the German military during World War II.

entropy: The fundamental measure of information. There are several closely-related definitions. The *Hartley-Shannon entropy* of a message source is the base-2 logarithm of the number of possible messages; it applies when all of the possible messages are equally likely. When the messages differ in probability, then the *Shannon entropy* is used. In both cases, Shannon’s first fundamental theorem tells us that the entropy equals the minimum average number of bits needed to represent the message in a binary code. *Thermodynamic entropy*, as discovered by Rudolf Clausius and Ludwig Boltzmann in the 19th century, is simply proportional to the information entropy of the unknown microstate of a macroscopic system. Some modern departures from Shannon’s information theory have their own revised

definitions of entropy. The *algorithmic entropy* of Andrey Kolmogorov and Gregory Chaitin is the length of the shortest computer program that can produce a particular binary string as its output. In quantum mechanics, the *von Neumann entropy* (formulated by John von Neumann) takes into account that “nearly” quantum states are not fully distinguishable.

error catastrophe: Condition for the preservation of the genetic information in a self-replicating molecule, first formulated by Manfred Eigen. If the genome length multiplied by the error rate is much larger than 1, the genetic information stored in the molecule will eventually be lost in the population. This means that for a given error rate, there is a maximum stable length for a genome.

flip-flop: A combination of Boolean logic gates with a “feedback” from output to input, which can serve as a simple memory unit for 1 bit.

Fourier transform: A way to mathematically represent a signal that varies in time or space as a combination of different frequencies, discovered by Joseph Fourier in 1822. A basic tool in perceptual coding of both audio and image data and important for understanding the bandwidth of an analog signal.

frequency analysis: A technique, first described in the 9th century by the Arab philosopher Al-Kindi, for breaking a substitution cipher by analyzing the frequency with which various letters appear.

full adder: A combination of Boolean logic gates that computes the sum of three 1-bit inputs and produces the answer as a 2-bit binary number. Because the full adder can include the results of a “carry” operation, a cascade of these devices can accomplish the addition of two binary numbers of any specified length.

gain: A measure of the geometric advantage of a directional antenna: the ratio of the directed signal power to the power of a signal that is spread equally in all directions. (A complementary definition describes the gain of a receiving antenna.) A parabolic high-gain antenna can achieve gain values of thousands or millions of times.

geometric mean: The square root of the product of two numbers or, for N numbers, the N^{th} root of their product. The average growth rate of an investment or debt is just the geometric mean of the annual growth rates over its lifetime.

half-adder: A combination of an AND gate and an XOR gate that computes the sum of two 1-bit inputs and produces the answer as a 2-bit binary number. The half-adder is a component part of a full adder.

Hamming distance: Introduced by information theorist Richard Hamming in 1950, a measure of the separation between two codewords in an error-correcting code, equal to the number of places in which they differ. For example, the English words GROVE and GRAVE, which differ in just one letter, have a Hamming distance of 1. If all the codewords of a code are separated by a Hamming distance of at least 3, then any single error can be corrected, because the resulting codeword will be “closer” to its original than to any other. With a greater minimum Hamming distance between codewords, even more errors can be corrected.

holographic principle: A speculative principle of physics based on the work of Jacob Bekenstein and developed by Gerard t’ Hooft, positing that the information contained in any region of space can be equally regarded as existing on the boundary surface of that region—the interior of a region is a kind of “hologram” produced by the surface.

Huffman code: A prefix-free binary code that is the most efficient code for a given set of message probabilities. Devised by MIT graduate student David Huffman in 1952.

interleaving: A method of defending against bursts of errors by splitting each codeword up and distributing across a large segment of a data stream. Audio CDs use both Reed-Solomon codes and interleaving to make them very error-resistant.

Johnson-Nyquist noise: Random electrical noise in a circuit produced by the thermal motion of electrons in it. Discovered by Bert Johnson and Harry Nyquist at Bell Labs in 1926. Extremely sensitive radio receivers, such as

those used by radio telescopes, are often cooled to a low temperature to reduce this internal source of interference.

JPEG: A flexible and highly efficient data-compression format for many kinds of image data, introduced by the Joint Photographic Experts Group in the 1990s. There is relatively little noticeable loss of image quality except for cartoons and diagrams involving fine lines and lettering, which tend to be surrounded by unwanted “artifacts” in the coded picture.

Kerckhoffs’s principle: The practical rule, first formulated by Auguste Kerckhoffs, that the designer of a cryptographic system should assume that any adversary knows in a general way the type of system that is used. Only the key information is secret. Later, Claude Shannon reformulated this rule as: “The enemy knows the system.”

key (cryptographic): In a cipher system, shared information between sender and receiver that is not shared by the eavesdropper. The secrecy of the key guarantees the privacy of the communication. One measure of the security of the system is the entropy of the key; the larger the entropy, the more secure the system.

long-term potentiation: The process by which connections between neurons are strengthened by use. This is the physical basis for long-term memory. The fact that the process of “reading” a memory is the same as the process of “writing” a memory means that memory is not a simple record of events; our memories can be changed over time.

macrostate: The condition of a large-scale thermodynamic system, characterized by a few large-scale variables, such as energy, volume, temperature, and so on. See also **microstate**.

Maxwell’s demon: A thought experiment in thermodynamics by James Clerk Maxwell, in which a demon acquires and uses detailed molecule-by-molecule information to reduce the entropy of a system, such as a container of gas. However, the second law of thermodynamics is not contradicted. Once the demon erases the vast quantities of data it has acquired (a necessary step for it to continue to operate), it will, by Landauer’s principle,

create enough waste heat in its environment to ensure that the total entropy will not decrease.

microstate: The detailed condition of a thermodynamic system, characterized by the exact state of every molecule it contains. This is a vast amount of information, even for a relatively small system, and we generally possess almost none of it. Thermodynamic entropy is essentially the amount of microstate information we lack. By the principle of microstate information, the distinction between microstates is preserved as the system evolves in time.

monogamy: The exclusive nature of the quantum information relationship known as *entanglement*. If quantum systems A and B are entangled, then no third system, C, can share in this exclusive relationship. The monogamy of entanglement is the basis for the security of quantum cryptography.

Moore's law: The observation by computer engineer Gordon Moore that the power of information-processing devices increases exponentially over time. The exact law is variously stated; one form says that the number of transistors on a single processor chip roughly doubles every two years.

Morse Code: The telegraph code devised by Samuel F. B. Morse in the 1840s. (The familiar modern version is a later revision of Morse's original.) Letters are represented by sequences of short and long electrical pulses (dots and dashes), and shorter sequences are used for more common letters. Thus, E is a single dot, while Z is dash-dash-dot-dot.

MP3: An audio data-compression format, originally developed as part of a system for data compression of video. (*MP* stands for "moving picture.") MP3 takes advantage of the masking phenomenon of human audio perception to greatly reduce the number of bits needed to transmit music or speech. For instance, 10 megabytes of uncompressed musical sound on an audio CD might occupy only 1 megabyte in MP3 format.

MPEG: A group of data-compression formats for video data, created by the Moving Picture Experts Group beginning in the late 1980s. In addition to techniques of image compression, the MPEG formats take advantage of the

fact that successive frames of a video usually have only small differences. One form of MPEG-4, also known as the H.264 format, is the usual standard for streaming video and Blu-Ray disks.

mutual information: The amount of information that is conveyed by a noisy channel. Mathematically, the difference between the entropy of the input and the conditional entropy of the input given the output. According to Claude Shannon's second fundamental theorem, the maximum of the mutual information for a channel is the channel capacity.

neuron: The basic unit of the brain for information processing and communication. Each neuron consists of a main body called the *soma* and a long fiber called an *axon*. Connections from other neurons induce the soma to produce nerve impulses, which are transmitted down the axon and influence other neurons connected at its end.

Ockham's razor: The logical principle, formulated by the 14th-century English philosopher William of Ockham, stating that the simplest theory is the one most likely to be true. Centuries after Ockham, Ray Solomonoff reformulated the principle using "monkey probabilities" from algorithmic information theory.

omega: The probability that a randomly generated program on a universal computing machine will eventually halt when it is carried out. Since the halting problem is uncomputable, so is the value of omega. Gregory Chaitin, who first described this number, has shown that omega contains the answers to all solvable mathematical problems, expressed in maximally compressed form.

one-time pad: An unbreakable cryptographic system in which shared key information is used only once and in which the essential problem is the secure distribution of the secret key information.

P and NP: Two classes of mathematical problems, expressing different computational complexity types. Problems in class P are reasonably easy to solve in both directions, but problems in class NP are easy in only one direction. The question of whether class NP includes more problems than class P is one of the greatest unsolved problems in mathematics.

perceptual coding: A powerful strategy for compression of media data, such as audio, still images, and video, by making a faithful representation of the subjective experience of the data, while losing information in ways that can be entirely unnoticeable. Pioneered by Manfred Schroeder in the 1970s.

phylogenetic tree: A tree structure representing how information flows from an original to generations of offspring, which can often be reconstructed by examining the copying errors that occur as the information is passed along.

plaintext: In a cipher system, the uncoded “plain language” message whose secrecy is to be protected.

prefix-free code: A code in which no codeword is the initial segment of any other codeword. Thus, if one codeword is 011, no other codeword can begin with 011... .

program: A data file that can also serve as a set of instructions for a computer.

pseudocode: An informal description of a computer program, readable by humans and containing all the essential details of an actual program.

public-key cryptography: A revolutionary method of cryptography that uses separate encryption and decryption keys. The encryption key can be published, but only the designated receiver has the decryption key. This completely avoids the problem of key distribution. In principle, it is always possible to deduce the decryption key from the published encryption key, but in practice, the necessary computation (factoring large integers into their prime factors) may be too difficult to accomplish.

quantization: The conversion of an analog quantity (such as a voltage) to a digital quantity, one with only a finite number of representable values. For example, the sample values for sound in an audio CD are represented by 16 bits, giving 65,536 possible sound levels at each moment.

quantum entanglement: A monogamous information relationship between quantum particles, in which neither particle by itself is in a definite quantum state, but the two together do have a definite quantum state.

quantum key distribution: An entirely practical and commercially available basic protocol of quantum cryptography by which a reliably secret random key can be shared between two parties. The best techniques use quantum entanglement. Any attempt by an eavesdropper to determine the key will necessarily disturb the shared quantum information in a way that the two parties will surely detect, leading them to reject the key they construct as insecure.

quantum no-cloning theorem: The information stored in a quantum system cannot be perfectly copied by any physical process. This fact, first proved in 1982, is one of the fundamental differences between quantum information and classical (Shannon) information.

qubit: The basic unit of quantum information; the information contained in a quantum system with just two distinguishable states, such as a photon polarization. In addition to 0 and 1 states, qubits can exist in many different quantum superposition states or in entangled states with other qubits. According to the quantum version of Claude Shannon's first fundamental theorem, the von Neumann entropy of a quantum information source equals the minimum number of qubits necessary to transfer the output of the source.

rate coding: A neural code in which the information is represented by the rate of the electrochemical pulses that travel down the axon. It has been known since the 1920s that at least some neural information is encoded in this way. Because the pulses occur somewhat at random, the capacity of this kind of code is limited by shot noise to a few bits per second. See also **temporal coding**.

redundancy: The property of language that uses many more bits (measured by the number of letters) to represent information than strictly necessary. This gives natural languages the property of error correction, but it also provides the basic vulnerability of a cipher system to cryptanalysis.

Reed-Solomon code: A powerful error-correcting code based on polynomial equations, invented by Irving Reed and Gustave Solomon. Used especially in situations where possible errors may occur in bursts, with several successive errors occurring together. Audio CDs use both Reed-Solomon codes and interleaving.

RNA: Ribonucleic acid, a sort of one-sided cousin to DNA. *Messenger RNA*, or mRNA, carries information from DNA to the ribosomes, where it is used as template for protein construction. *Transfer RNA*, or tRNA, is a short, looped form of the molecule that is attached to amino acids and helps to recognize the codon sequences in mRNA. RNA molecules can also act as biochemical catalysts, called *ribozymes*.

rolling code: Method used by many remote key systems of generating a new key sequence for each press of the key button. Key sequences are generated by a fixed mathematical algorithm, which could in principle be deduced by monitoring repeated key signals. See also **one-time pad**.

run-length encoding: An efficient technique for encoding simple images, in which the number of contiguous identical pixels is given. Not very efficient for photographs or other images with complex textures and fine variations in shading.

sampling: The representation of a continuous signal (such as a sound wave) by a series of discrete values. According to the Nyquist-Shannon sampling theorem, a sampling rate of twice the signal bandwidth is both necessary and sufficient to represent the signal faithfully. Because human hearing reaches to a frequency of only around 20,000 Hz, audio CDs sample the sound at 44,100 samples per second. This is enough to reproduce the audible sound.

Shannon's maxim: See **Kerckhoffs's principle**.

sidebands: The spread of radio energy to frequencies on either side of the basic carrier frequency, produced when information is added in the form of AM or FM radio signals (amplitude or frequency modulation). The presence of sidebands determines the bandwidth of the radio signal.

signal-to-noise ratio: Also known as SNR, the ratio of the power of a signal to the power of the interfering noise. This is often described logarithmically, using decibels. Along with the signal bandwidth, the signal-to-noise ratio limits the information rate of an analog signal, according to the Hartley-Shannon entropy.

SIGSALY: An encrypted and perfectly secure telephone system used for high-level communications during World War II, in which identical phonograph records of noise were added to and subtracted from a digitally encoded voice signal.

substitution cipher: A cipher system in which each letter of the plaintext is replaced by another letter or symbol according to a constant rule. Substitution ciphers are relatively easy to break by frequency analysis, if the ciphertext is long enough.

surprise: The novelty of a random event, measured in bits, equal to the logarithm of the reciprocal of its probability. If event A is half as likely as event B, its surprise is 1 bit greater. The Shannon entropy of an information source is the average surprise of its messages.

temporal coding: A neural code in which the information is represented by the exact timing of the electrochemical pulses that travel down the axon. Temporal coding allows a greater information capacity than simple **rate coding**. Several lines of indirect evidence suggest that this is part of the neural code in at least some parts of the brain.

transistor: A solid-state electrical component that allows one electrical signal to control another, invented in 1948 by John Bardeen, Walter Brattain, and William Shockley, all of Bell Labs. Basic logic gates can be built out of a few transistors.

Turing machine: An extremely simple but universal computing machine described by Alan Turing in 1937 as a very small and simple device that moves forward and backward along an unlimited data tape, reading and writing symbols according to a simple rule. Such a machine can be shown to

be equivalent to an extremely sophisticated computer in that the two types can simulate each other and, thus, perform exactly the same computations.

unicity distance: The amount of ciphertext data required to uniquely specify the original plaintext. For a simple substitution cipher, Shannon estimated a unicity distance of 28 letters. Thus, any enciphered message much longer than this should be breakable by cryptanalysis.

Vigenère cipher: A cipher system using a continually shifting set of substitution ciphers, determined by a repeated key phrase of any fixed length. Known as *le chiffre indéchiffrable* (“the indecipherable cipher”) because it cannot be broken by simple frequency analysis, but Charles Babbage invented a way to break it.

von Neumann machine: A hypothetical robot able to assemble a copy of itself from a supply of basic components. John Von Neumann showed that it had to contain a set of instructions in its memory that was both expressed (in the building process) and copied as data (into the memory of the “offspring” machine).

XOR: The “exclusive OR” operation in Boolean logic. When both input bits agree (00 or 11), then the output is 0; when they disagree (01 or 10), the output is 1.

Zipf’s law: An empirical fact about language first noted by the linguist George Kingsley Zipf in 1935. Simply stated, in any long sample of English text, the N^{th} most common word is about N times less likely than the most common word. Similar regularities occur in other languages.

Bibliography

PRINT RESOURCES

Aaronson, Scott. *Quantum Computing since Democritus*. Cambridge, 2013. Scott Aaronson is one of the most brilliant researchers in quantum information today. He is also a gifted communicator with a rollicking sense of humor. This book is aimed at the nonspecialist, but it voyages into some deep intellectual waters.

Aczel, Amir D. *Entanglement*. Plume, 2001. This is a good, accessible book by a fine writer on the general subject of quantum entanglement. Because the book was written as long ago as 2001, however, the term *monogamy* does not appear in the index.

Atkins, Peter. *The Laws of Thermodynamics: A Very Short Introduction*. Oxford, 2010. This book is just what it claims to be: an introductory survey of the laws of thermodynamics, written by a former professor of chemistry at Oxford. Like Atkins's other popular science books, this one is exceedingly well put together and full of beautiful insights, even for the practicing thermodynamicist.

Avery, John Scales. *Information Theory and Evolution*. 2nd ed. World Scientific, 2012. Written by an eminent theoretical chemist, this book is an ambitious attempt to bring together ideas of information science to understand the essential issues of biological and cultural evolution, as well as the implications of new technologies. Avery does a good job at the difficult task of combining technical sophistication with accessibility.

Benford, Gregory. *Deep Time*. HarperCollins, 1999. Both a noted physicist and a noted science fiction writer, Benford was on one of the scientific teams studying the 10,000-year marker problem for the Waste Isolation Pilot Plant and helped design spacecraft-borne messages to unknown recipients in deep space. His book is an intelligent and imaginative discussion of the many dimensions of the problem of transmitting information, both human and biological, into the far-distant future.

Cairns-Smith, A. G. *Seven Clues to the Origin of Life*. Canto, 1990. Cairns-Smith explains his fascinating clay crystal idea about the earliest life on Earth. Even apart from his own theories, the book is an elegant and entertaining account of the essential issues in the problem of life's origin.

Cavalli-Sforza, Luigi Luca, and Francesco Cavalli-Sforza. *The Great Human Diasporas*. Perseus, 1996. In this fascinating book, Cavalli-Sforza and his son Francesco explore how the phylogenetic tree of humanity has been reconstructed in recent years. One of the fascinating aspects of this quest has been the way in which the genetic and linguistic evidence complement and reinforce each other.

Chadwick, John. *The Decipherment of Linear B*. Canto, 1990. Michael Ventris died in 1956, too early to write his own popular account of the breakthrough that unlocked the ancient Mycenaean inscriptions from Crete. But his collaborator, philologist John Chadwick, gives us a book that is at once serious and lively, full of insights into the deep cryptanalysis required to decipher an ancient script.

Chaitin, Gregory J. *Thinking about Gödel and Turing*. World Scientific, 2007. A gathering of essays by Chaitin (including three *Scientific American* articles), this book gives a survey of algorithmic information ideas and their connection to meta-mathematics. Because the contributions range over several decades, there is some inevitable redundancy and changes in perspective; nevertheless, this is a fine collection by a remarkable thinker.

Cook, Perry, ed. *Music, Cognition and Computerized Sound: An Introduction to Psychoacoustics*. MIT, 2001. Psychoacoustics is the study of how human beings perceive audio data. This collection of introductory chapters by different authors—some of them notable information scientists—provides a useful overview.

Davies, Paul. *The Fifth Miracle*. Penguin, 1998. This excellent book about the origin of life comes from an eminent physicist who has brought his insight and training to bear on a remarkably wide range of scientific questions, from cosmology to cancer. He is also an award-winning writer and broadcaster on scientific subjects.

Doody, Dave. *Basics of Space Flight*. Bluroof Press, 2011. Doody, a NASA engineer specializing in spacecraft operations, originally wrote this as an online primer on all aspects of space missions; later, it was published as an excellent book. Doody's discussions of telecommunications and the Deep Space Network will be particularly rewarding to students of this course.

Gleick, James. *The Information: A History, a Theory, a Flood*. In this book, one of the best science writers of the modern era takes on the information revolution. (Gleick's previous books included bestsellers on chaos theory and the life of physicist Richard Feynman.) As one would expect from the author, this book contains a rich trove of insights and delightful examples, from the "talking drums" of West Africa to the social dynamics of Wikipedia.

Hamming, Richard. *Coding and Information Theory*. Prentice Hall, 1980. This is my favorite of the many mathematical textbooks on information theory, and it is written by one of the discoverers of error-correcting codes. Though advances in the coding field have rendered it slightly out of date, it remains valuable for its clear approach and clean mathematical arguments.

Hinsley, F. H., and Alan Stripp, eds. *Codebreakers: The Inside Story of Bletchley Park*. Oxford, 1993. This is a fascinating account of the British effort to break the German Enigma code during World War II, written by the participants. It covers both the technical side and the human side: What was it like to work endless hours on a super-secret project that affected the whole outcome of the war?

Hofstadter, Douglas B. *Gödel, Escher, Bach: An Eternal Golden Braid*. Vintage Books, 1980. This is a book about which superlatives seem appropriate. It is one of the most delightful and original books about mathematics ever written, a winner of both a Pulitzer Prize and a National Book Award. Fantastical dialogues, fugues, fractals, formal mathematical systems, and much more crowd its pages. It also remains one of the deepest discussions of Turing machines and the computation theory ever written for a lay audience.

Leff, Harvey S., and Andrew F. Rex, eds. *Maxwell's Demon 2*. Institute of Physics, 2003. Following the publication of a previous (and also excellent) historical collection of papers about Maxwell's demon in 1990, Professors

Leff and Rex realized that a second volume was almost immediately needed to reflect more recent discoveries and insights. Original papers by Szilard, Brillouin, Landauer, Bennett, Zurek, Lloyd, and many others give a rich, comprehensive view of the modern subject of Maxwell's demon. Some papers are technical; others are accessible to the lay reader. The editors' 40-page introductory survey and the extensive bibliography make this an indispensable book for the information physicist.

Loepp, Susan, and William K. Wootters. *Protecting Information*. Cambridge, 2006. This undergraduate textbook by two professors from Williams College is at a somewhat higher mathematical level than some other items in this bibliography. The book gives a clear and sophisticated introduction to cryptography, error-correcting codes, quantum cryptography, and quantum computation.

Petzold, Charles. *The Annotated Turing*. Wiley, 2008. Petzold, a well-known computer scientist and writer about computer programming, undertakes a detailed explication of Alan Turing's fundamental 1936 paper that introduced the Turing machine. Along the way, he presents intriguing historical, biographical, and technological commentary. Though the book is somewhat mathematical, it aims to make the mathematical ideas as widely accessible as possible.

Pierce, John R. *An Introduction to Information Theory: Symbols, Signals and Noise*. Dover, 1980. This is an updated edition of one of the first—and still one of the best—accessible introductions to Shannon's theory. Pierce was an engineer at Bell Labs when information theory first began, and he made fundamental contributions to its development. Later, he was on the faculties of Caltech and Stanford and served for a time as chief engineer at the Jet Propulsion Laboratory.

Poundstone, William. *Fortune's Formula*. Hill and Wang, 2005. Poundstone cannot write a boring book, and this is no exception to that rule. Not only is this a superb popular account of John Kelly's discoveries and the ideas of mathematical finance, but it is also a delightful tale full of vivid characters ranging from mathematicians, to gangsters, to economists, to physicists, to financiers.

Renyi, Alfred. *A Diary on Information Theory*. Wiley, 1987. Hungarian mathematician Alfred Renyi made profound contributions to probability and information theory. In this quirky and charming book, he pens the imaginary diary of a mathematics student coming to grips with Shannon's theory for the first time. (Later chapters touch on a wide variety of mathematical ideas.)

Sagan, Carl, ed. *Communication with Extraterrestrial Intelligence*. MIT, 1975. This book is the proceedings of a joint meeting between scientists from the United States and the USSR to discuss the technical and scientific issues of communicating with extraterrestrials. The discussions are as fascinating as the formal presentations. The conference took place after Drake's Project Ozma and the Arecibo message and set the agenda for SETI research to this day.

Schrödinger, Erwin. *What Is Life?* Cambridge, 1967. This reprint of Schrödinger's classic 1944 lectures on the physical nature of life also includes the essays "Mind and Matter" and "Autobiographical Sketches." Schrödinger's prose is lucid, and his arguments were extraordinarily influential for scientists of many disciplines as they unraveled the basic information system of living things.

Seife, Charles. *Decoding the Universe*. Viking, 2006. This is an excellent book on Shannon's information theory and the adoption of its ideas into biology and physics. Seife, an accomplished science journalist, knows his territory well and chooses his topics and examples to great effect. Even an expert can learn something from his lucid explanations and appealing illustrations.

Shannon, Claude, and Warren Weaver. *The Mathematical Theory of Communication*. University of Illinois, 1971. This book collects Shannon's long, revolutionary information paper from the *Bell System Technical Journal* with a shorter, nontechnical essay by Warren Weaver. Even though Shannon's paper is the foundation of a highly mathematical subject, it is so lucidly written that a lay reader will gain much from it.

Siegfried, Tom. *The Bit and the Pendulum*. Wiley, 2000. Siegfried is a truly excellent science journalist and served for many years as the editor of *Science News*. His book is a highly readable account of the "invasion" of information ideas into physics and biology in the 1980s and 1990s.

Singh, Simon. *The Code Book*. Doubleday, 1999. Possibly the best popular history of cryptography, Singh's book is full of fascinating stories and brilliant explanations. His careful description of the German Enigma system in World War II—including the way that Turing's "loop" approach to cryptanalysis bypassed the plugboard settings—is first-rate. The book also explores the Vigenère cipher, the decipherment of Linear B, public-key cryptography, and quantum cryptography. Unaccountably, Singh does not mention Shannon's information-based theory of cryptography and cryptanalysis, but this is a small flaw in an otherwise excellent book.

Von Baeyer, Hans Christian. *Information: The New Language of Science*. Harvard, 2004. Von Baeyer is a professor of physics at The College of William and Mary and has written a number of excellent books on physics for the lay reader. His approach to the science of information is that of a physicist who is particularly interested in "really big questions"; thus, the student of this course will find much that is familiar!

Wheeler, John Archibald, and Kenneth W. Ford. *Geons, Black Holes and Quantum Foam: A Life in Physics*. Norton, 2000. John Wheeler's physics memoir, written with Ken Ford, is a wonderful introduction to his amazing career and unique style of thinking. Students of this course will be particularly interested in the later chapters, in which Wheeler discusses his quest for the information basis of physics: "it from bit."

ONLINE RESOURCES

The Dasher Project. www.inference.phy.cam.ac.uk/dasher/. Provides more information about Dasher and free downloadable software for most operating systems.

Golomb, Solomon W., et al. "Claude Elwood Shannon (1916–2001)." *Notices of the AMS* 49 (2002): 8–16. www-isl.stanford.edu/~cover/papers/paper117.pdf.

Kelly, John. "A New Interpretation of Information Rate." *Bell System Technical Journal* 35 (1956): 917–926. Some libraries permit online access to Kelly's paper.

The Logic Lab. www.neuroproductions.be/logic-lab/. A fun and enlightening online program for experimenting with logic gates and flip-flop circuits.

Shannon, Claude. "Communication Theory of Secrecy Systems." *Bell System Technical Journal* 28-4 (1949): 656–715. pages.cs.wisc.edu/~rist/642-spring-2014/shannon-secrecy.pdf.

Image Credits

page no. 5	© Photos.com/Thinkstock.
11	© 3quarks/Depositphotos.com
17	© Kodamatobi/iStock/Thinkstock.
18	© Library of Congress Prints and Photographs Division, LC-USZ62-66023.
18	© Jitze Couperus/Flickr/CC BY 2.0.
19	© Science Museum London/flickr/CC BY-SA 2.0.
21	© Ineuw/Wikimedia Commons/Public Domain.
29	© scyther5/iStock/Thinkstock.
36	© cookie_cutter/iStock/Thinkstock.
46	© Library of Congress Prints and Photographs Division, LC-USZ62-2188.
46	© Studio-Annika/iStock/Thinkstock.
106	© Stocktrek Images/Thinkstock.
106	© NASA.
110	© MarioGuti/iStock/Thinkstock.
120	© Central Intelligence Agency.
130	© Daderot/Wikimedia Commons/Public Domain.
141	© Archive.org.
162	© United States Geological Survey.
194	© Depositphotos.com/Nicku.
296	© vintagedept/flickr/CC BY 2.0.
300	© Isaac Ruiz Santana/iStock/Thinkstock.